 ISO/IEC JTC 1/SC 29/WG 03 N 1689

**ISO/IEC JTC 1/SC 29/WG 03**

**MPEG Systems   
Convenorship: KATS (Korea, Republic of)**

**Document type:** Output Document

**Title: Technologies under Consideration for Dynamic Adaptive Streaming over HTTP 23009, Parts 1, 3, 4, 5, 6 and 8**

**Status:** Approved

**Date of document:** 2026-01-23

**Source:** ISO/IEC JTC 1/SC 29/WG 03

**No. of pages:** 60 (with cover page)

**Email of Convenor:** young.L@samsung.com

**Committee URL:** <https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg3>

**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 03 MPEG SYSTEMS**

**ISO/IEC JTC 1/SC 29/WG 03 N1689**

**Online, Jan 2026**

|  |  |
| --- | --- |
| **Source** | **MPEG Systems WG** |
| **Status** | **Output** |
| **Title** | **Technologies under Consideration for Dynamic Adaptive Streaming over HTTP 23009, Parts 1, 3, 4, 5, 6 and 8** |
| **Editor** | **Iraj Sodagar** |
| **Serial Number** | **25878** |

This Technology under Consideration document collects candidate technologies for inclusion into new amendments of ISO/IEC 23009 Dynamic Adaptive Streaming over HTTP.

Note that any section that is not indicated otherwise relates to Part 1.

Contents

[1 Ad insertion 6](#_Toc220055475)

[2 Failover in multi-origin linear deployments (m54725) 7](#_Toc220055476)

[2.1 Introduction 7](#_Toc220055477)

[2.2 Proposal 7](#_Toc220055478)

[2.2.1 Approach 7](#_Toc220055479)

[2.2.2 Proposed syntax and semantics 8](#_Toc220055480)

[3 Extensions for Service Description (m56093 and m67854) 9](#_Toc220055481)

[3.1 Extensions based on 3GPP 5GMS (m56093) 9](#_Toc220055482)

[3.1.1 Introduction 9](#_Toc220055483)

[3.1.2 Background 9](#_Toc220055484)

[3.1.3 Live services at different scale 14](#_Toc220055485)

[3.1.4 Scalability 14](#_Toc220055486)

[3.1.5 Consistent quality 14](#_Toc220055487)

[3.1.6 Deployment Architectures 15](#_Toc220055488)

[3.1.7 Operation Point – Establishment and Monitoring 16](#_Toc220055489)

[3.1.8 Key Issues for DASH 16](#_Toc220055490)

[3.1.9 Proposed Updates Service Description 16](#_Toc220055491)

[K3.X Operating Mode 16](#_Toc220055492)

[3.2 Extending the Service Description (m67854) 17](#_Toc220055493)

[3.2.1 Introduction 17](#_Toc220055494)

[3.2.2 Service Description Today 17](#_Toc220055495)

[3.2.3 DASH.js Settings 18](#_Toc220055496)

[3.2.4 Issues 19](#_Toc220055497)

[3.2.5 Solution for 1: Extend Service Description with additional parameters 19](#_Toc220055498)

[3.2.6 Solution for 2: Networking Notifications 19](#_Toc220055499)

[4 MSE implementation of inband events (m56684) 22](#_Toc220055500)

[5 Clarifying pre-roll signaling for seamless content splicing across MPEG-DASH Periods (m56890) 23](#_Toc220055501)

[5.1 Introduction 23](#_Toc220055502)

[5.2 Related contributions 23](#_Toc220055503)

[5.3 Discussion 23](#_Toc220055504)

[5.3.1 Pre-roll context 23](#_Toc220055505)

[5.4 Use-cases 23](#_Toc220055506)

[5.4.1 Ad insertion using multiple Periods 23](#_Toc220055507)

[5.4.2 Open-GOPs splicing 24](#_Toc220055508)

[5.5 Discussions 24](#_Toc220055509)

[5.6 Conclusion 25](#_Toc220055510)

[6 Enabling CMCD beaconing (m65127) 26](#_Toc220055511)

[6.1 Introduction 26](#_Toc220055512)

[6.2 Notes from MPEG#144 27](#_Toc220055513)

[7 Signaling maximum segment size (m65860) 29](#_Toc220055514)

[7.1 Introduction 29](#_Toc220055515)

[7.2 Proposal 29](#_Toc220055516)

[7.2.1 Definition 29](#_Toc220055517)

[7.2.2 Operational model 29](#_Toc220055518)

[8 Extending copyright license signaling (m66512) 31](#_Toc220055519)

[8.1 Introduction 31](#_Toc220055520)

[8.2 Background 31](#_Toc220055521)

[8.3 Use case and justification 32](#_Toc220055522)

[8.4 Proposed solution 32](#_Toc220055523)

[8.4.1 Alternative solution 33](#_Toc220055524)

[8.5 Notes from MPEG#145 34](#_Toc220055525)

[9 DASH events for C2PA metadata signaling (m72350) 35](#_Toc220055526)

[9.1 Abstract 35](#_Toc220055527)

[9.2 Introduction 35](#_Toc220055528)

[9.3 C2PA based streaming PoC 35](#_Toc220055529)

[9.4 Proposed modifications to ISO/IEC 23009-1 36](#_Toc220055530)

[9.5 Recommendations 37](#_Toc220055531)

[9.6 References 37](#_Toc220055532)

[10 Association of provenance and authenticity of multiple media representations in DASH manifest (m75658) 38](#_Toc220055533)

[10.1 Abstract 38](#_Toc220055534)

[10.2 Introduction 38](#_Toc220055535)

[10.3 Proposal 38](#_Toc220055536)

[12 Latency Reporting (m72386) 39](#_Toc220055537)

[12.1 Introduction 39](#_Toc220055538)

[12.2 Principles 39](#_Toc220055539)

[12.3 What does CMCD v2 define 40](#_Toc220055540)

[12.4 Proposal 40](#_Toc220055541)

[13 “urn:mpeg:dash:…” namespace assignments 41](#_Toc220055542)

[13.1 Abstract 41](#_Toc220055543)

[13.2 Introduction 41](#_Toc220055544)

[13.3 Proposal 41](#_Toc220055545)

[13.3.1 On urn:mpeg:dash:encryptionkey-server: 42](#_Toc220055546)

[13.4 Recommendations 42](#_Toc220055547)

[13.5 References 42](#_Toc220055548)

[14 CMCD V1 and V2 (m74567, m75722) 43](#_Toc220055549)

[14.1 Introduction 43](#_Toc220055550)

[14.2 Open Questions related to 6th edition 43](#_Toc220055551)

[K3.7 Client-Data Reporting Configuration 43](#_Toc220055552)

[K3.7 Client-Data Reporting Configuration 45](#_Toc220055553)

[14.3 What about CMCDv2 47](#_Toc220055554)

[14.3.1 Example Emissions 49](#_Toc220055555)

[**Transport semantics** 53](#_Toc220055556)

[**Example header & query formatting** 54](#_Toc220055557)

[15 Energy-aware Streaming (m75723) 55](#_Toc220055558)

[15.1 Introduction 55](#_Toc220055559)

[15.2 Functional description 55](#_Toc220055560)

[15.2.1 Overview 55](#_Toc220055561)

[15.2.2 Assignment and aggregation of energy usage 56](#_Toc220055562)

[15.2.3 Common repository of energy-related metrics 56](#_Toc220055563)

[15.2.4 In-band carriage of energy-related metrics 57](#_Toc220055564)

[15.2.5 Carriage of energy-related information in streaming presentation manifest 57](#_Toc220055565)

[15.2.6 Usage of energy information in media client or other functions 58](#_Toc220055566)

[15.3 Procedures 58](#_Toc220055567)

# Ad insertion

This clause was addressed by the alternative media presentation tool in the 6th edition.

# Failover in multi-origin linear deployments (m54725)

Note: The failover is already supported by the DASH spec. This clause provides additional ideas to handle the failover.

## Introduction

In many cases multiple origins are used to serve the same content. In particular, in a geo-redundancy case same channel is transcoded by different transcoders and published to different origins. In case one origin fails, the player can seamlessly or near-seamlessly switch to a different origin.

In the naïve implementation, the client will quit on an origin failure and start with a new URL pointing to a different origin. This solution is unacceptable, as it incurs a significant delay due to buffering, tearing down the current DRM session, getting a new URL from the back-end, downloading the MPD and initialization segments, reinitializing the decoder, requesting new DRM license, setting up the new DRM context, etc. The overall effect can be very noticeable, as the player will be down for e.g. 30 seconds. Moreover, an origin failure affecting hundreds of thousands or millions of concurrent sessions will create a “thundering herd” effect on the licensing servers as hundreds of thousands or millions of license requests will come at the same time, taxing the license server infrastructure.

DASH provides two mechanisms to reduce the downtime: Fallback MPD chaining and MPD reset.

Fallback MPD chaining provides the client with an alternative MPD it can use in case of a catastrophic failure such as failure to do normal MPD update. MPD reset tells the client that it has to switch to playback of a new MPD at a given time. In both cases, the player can finish playing its current buffer and switch to the new MPD. While this is an improvement over the naïve implementation, the DRM issue remains.

In case we continue viewing the same channel, there typically is no need to request a new license – if the viewer was authorized to view a channel from one origin, then he/she will be authorized to view it from a different origin. As long as the origins have matching encryption keys (even if `pssh` is different), there is no need to re-request them or do anything to the secure rendering pipeline.

A different issue is preserving user experience. For example, if the MPD offers multiple views of the same content, we want to continue watching the same view we were watching prior to the failover. In case two MPDs have same adaptation set IDs, we can switch to precisely the same view.

These types of continuity need to be somehow signaled to the player which currently cannot tell whether it is switching to a slate with an error message or to a very similar MPD with different segment URLs and difference in segment timing. This type of signaling is proposed below.

## Proposal

### Approach

We define several types of continuity:

1. Asset continuity: both MPDs show the same visual content and have approximately the same live point. It is hard to get to precisely same live point when the encoders are at significantly different geographic locations and same frame enters different transcoders at a different time due to speed of light.
2. Service continuity: the adaptation set and representation IDs are identical, so it is possible to simplify the switch logic to follow the currently playing IDs as opposed to starting from a “clean slate”. Additionally, the same events are synchronized (i.e., the events are the same in their semantics), which is important in terms of advertising logic and SCTE 35.
3. Content protection continuity: as long as same key IDs are used, there is no need to re-request the DRM license

Note that the structure here is not a perfect “onion structure” – content protection continuity requires asset continuity, but not service continuity.

We propose a new element to explicitly provide such signaling and explicitly call out different types of continuity.

### Proposed syntax and semantics

The Continuity element describes various aspects which are maintained across representations described by successive MPDs. This element is embedded inside a SupplementalProperty or EssentialProperty element with @schemeIdUri attribute values of urn:mpeg:dash:fallback:2016 or urn:mpeg:dash:reset:2016.

Table 25 — Semantics of Continuity element

| **Element or Attribute Name** | | | | **Use** | **Description** |
| --- | --- | --- | --- | --- | --- |
|  |  | Continuity | |  |  |
|  |  |  | @asset | OD  default=false | If true, content of the new presentation is perceptually identical to the content of the current presentation, i.e. the “live point” of one presentation is within 250ms from the other |
|  |  |  | @service | OD default=false | If true, adaptation set, representation, and sub-representation IDs are identical across the new and old representation |
|  |  |  | @contentProtection | OD default=false | If true, the DRM license delivered for segments in the old presentation, is still true for segments in the new presentation with same key IDs |
| **Key**  For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory  For elements: <minOccurs>..<maxOccurs> (N=unbounded)  Elements are bold; attributes are non-bold and preceded with an @. | | | | | |

# Extensions for Service Description (m56093 and m67854)

## Extensions based on 3GPP 5GMS (m56093)

### Introduction

This document provides proposed updates to Service Description. The background for this is the usage of Service Description as part of 5G Media Streaming.

### Background

3GPP has specified a system for **5G Media Streaming** that enables a mobile network operator to offer a level of service that goes beyond "best effort" over-the-top IP-based media streaming. 5G Media Streaming services offered by a **5GMS System** are provisioned by a third-party actor referred to as the **5GMS Application Provider** for use by an application running on the User Equipment (UE) referred to as a **5GMS-Aware Application**. The reference architecture and basic functional procedures are defined in TS 26.501 [X] and the detailed protocols are specified in TS 26.512 [Y]. The baseline video codecs and packaging standards that compliant UEs must support as a minimum are specified is TS 26.511 [Z].

In 3GPP Release 16, the scope of these specifications is restricted to **unicast media streaming** only. A **Content Hosting** capability is defined that resembles a Content Delivery Network (CDN). Later releases may add support for more complex media hosting and manipulation features. Ongoing Release 17 studies and normative work seek to add **multicast/broadcast** distribution mechanisms to the 5G System as well as **edge computing** capabilities.

The following high-level features are specified for 5G Media Streaming in Release 16. Each feature is optional and only available to a 5GMS-Aware Application if explicitly provisioned by a 5GMS Application Provider:

1. **Content Hosting.** This may be deployed inside the 5G Core network in the form of an Operator CDN. Alternatively, an external third-party CDN may be integrated into the 5G Media Streaming system.
2. **Media Consumption Reporting.** A random subset of 5GMS Clients can be configured to periodically report media session usage information to the 5GMS System.
3. **QoE Metrics Reporting.** A random subset of 5GMS Clients can be configured to periodically report Quality of Experience metrics to the 5GMS System. These may be relayed to the 5GMS Application Provider.
4. **Dynamic Network QoS Policies.** Specific network QoS policies are provisioned in advance, expressed as **Policy Templates**. During streaming sessions these Policy Templates can then be instantiated on demand by individual 5GMS Clients. The 5GMS Application Function negotiates with the Policy and Charging Function (PCF) in the 5G Core to apply the requested QoS policy to the relevant 5GMS packet flow.
5. **Network Assistance.** Two forms of assistance are currently defined. Neither requires any special configuration at the provisioning stage.
   1. The 5GMS Client can interrogate the network to find out what downlink network capacity is currently available to it. This can be used to influence the Media Player's choice of media representations to best ensure an uninterrupted streaming experience.
   2. The 5GMS Client can request a temporary "boost" to its network Quality of Service, for example to speed up a background download (network resources permitting).



Figure X Reference architecture for 5G Media Downlink Streaming (see TS 26.501 [X])

The reference architecture for 5G Media Streaming as shown in Figure X defines the following functions to support the abovementioned features:

* A **5GMS Application Function** deployed in the 5G Core or in an External Data Network that manages a 5GMS System. This logical function embodies the control plane aspects of the system, such as provisioning, configuration and reporting:
  + A 5GMS Application Provider provisions 5GMS functions using a RESTful HTTP-based provisioning interface at reference point **M1**.
  + Another RESTful HTTP-based configuration and reporting interface is exposed to 5GMS Clients at reference point **M5**.
* A **5GMS Application Server** deployed in the 5G Core or in an External Data Network that provides 5G Media Streaming services to 5GMS Clients. This logical function embodies the data plane aspects of the system that deal with media content:
  + Content is ingested from 5GMS Application Providers at reference point **M2**. Both push- and pull-based ingest methods are supported, based on HTTP.
  + Content is distributed to 5GMS Clients at reference point **M4** (after possible manipulation by the 5GMS Application Server function). Standard pull-based content retrieval protocols (e.g. DASH) are supported at this reference point.
* A **5GMS Client** deployed in the UE that consumes 5G Media Streaming services. The 3GPP specifications are silent on whether this logical function is realised as shared UE middleware components or provided piecemeal by individual applications.
  + A **Media Session Handler** subcomponent first retrieves its configuration ("Service Access Information") from the 5GMS Application Function at reference point **M5** and then uses this configuration information to activate and exploit the currently provisioned 5GMS features. The 5GMS-Aware Application controls the Media Session Handler via a UE-internal API defined at reference point **M6**. This reference point could, for example, be realised as a Javascript API in a web browser.
  + A **Media Player** subcomponent consumes media from the 5GMS Application Server at reference point **M4**. The 5GMS-Aware Application controls the Media Player via a UE-internal API defined at reference point **M7**. This reference point could also be realised as a Javascript API in a web browser, for example.

The basic procedures for 5G Media Streaming are shown in Figure Y.



Figure Y Basic procedures for 5G Media Downlink Streaming

According to TS 26.501 [X], Downlink Media Streaming provides the ability for content to be distributed using procedures and protocols defined by 5G Media Streaming as shown in Figure Z. The detailed procedures for the interfaces and APIs for 5G Media Streaming are defined in TS 26.512 [Y].



Figure Z <caption>

5G Media Streaming segment formats are defined based on the Common Media Application Format (CMAF) in ISO/IEC 23000-19 [A]. By using this format, 5G Media Streaming is compatible with a broad set of segment-based streaming protocols including Dynamic Streaming over HTTP (DASH) and HTTP Live Streaming (HLS). For example, ISO/IEC 23009-1 [B] defines a detailed DASH profile for delivering CMAF content within a DASH Media Presentation using a converged format for segmented media content.

5GMS media profiles for video, audio and subtitles based on the general constraints of ISO/IEC 23000-19 [A] are defined in TS 26.511 [Y]. However, 5G Downlink Media Streaming is not restricted to the media profiles defined in [Y]. Any CMAF media profile, for example for codecs defined in DVB specifications, may be used and distributed within 5G Downlink Media Streaming.

A more detailed client-centric approach is shown in Figure 1.

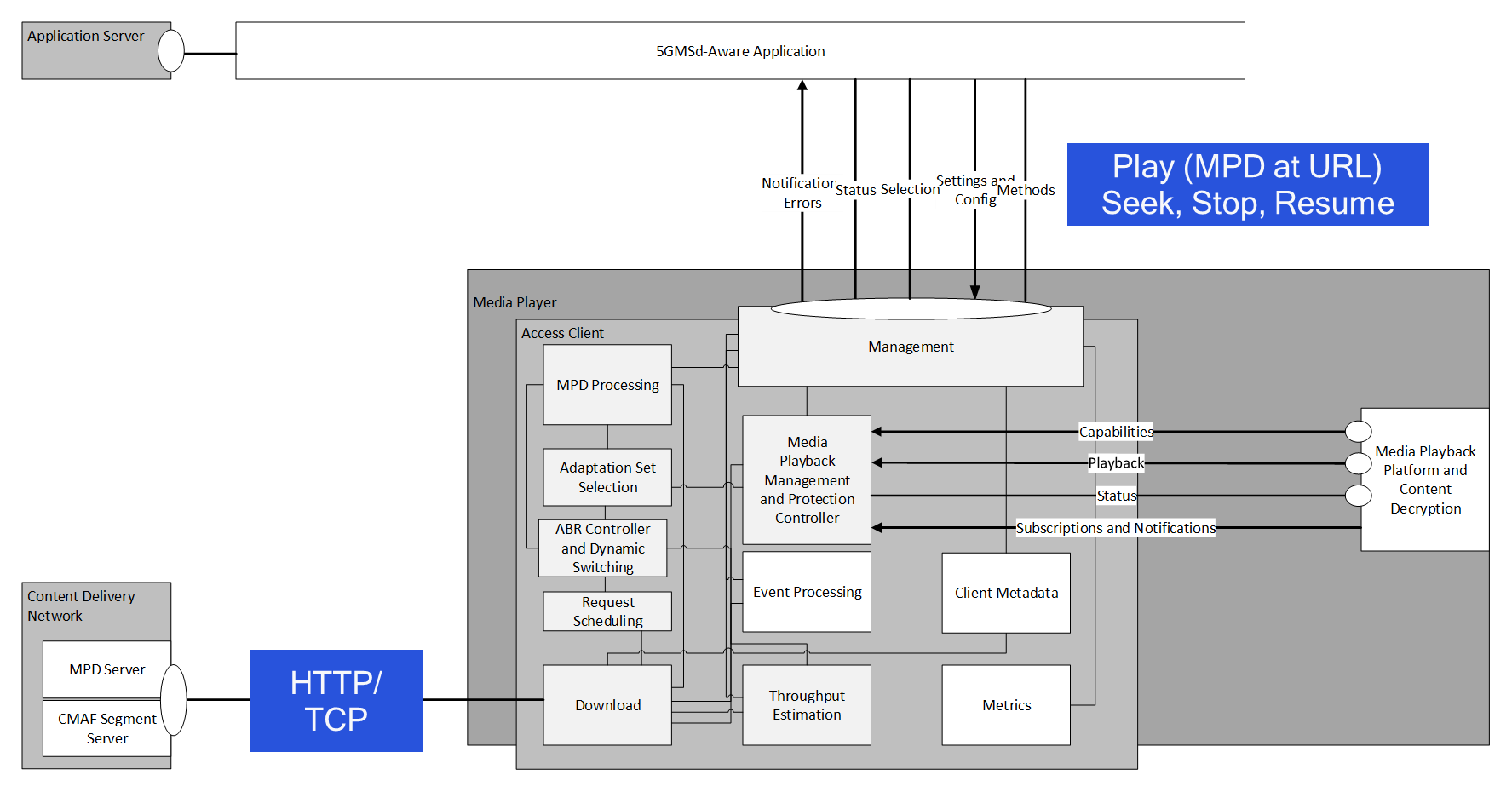


Figure 1 Client centric architecture

An important concept in 5G Media Streaming are Service Operation points and policy templates. The details are shown in Figure 2. An overview is provided in the following.

* Service Operation Points
  + Service Operation points define **long lived profiles** that will be used by streaming sessions as references.
  + Policy templates represent **long term agreements made between the AP and the MNO**.
  + Filtering: AP can limit what traffic and which users are allowed to use a specific policy template (FQDN-based)
  + Streaming session uses at most one of the allowed policy templates at any point in time
  + MSH may pre-cache or retrieve periodically or on request the list of allowed operation points for a specific AP
* Option 1: Define each Service Operation Point as a Slice
  + concept of Network Slice as a Service (NSaaS) is defined in TS 28.530 [10].
  + NSaaS can be offered by an MNO to third-party providers in the form of a service.
  + UE establishes or modifies the PDU session that will be used for the traffic based on URSP rules
  + Alternatively, network may also trigger the establishment or modification of the PDU session
* Option 2: Define each Service Operation Point as a QoS Flow
  + Flow description(s) of the transport session established in the step (see TS 23.502 [3]), e.g. the 5-tuple

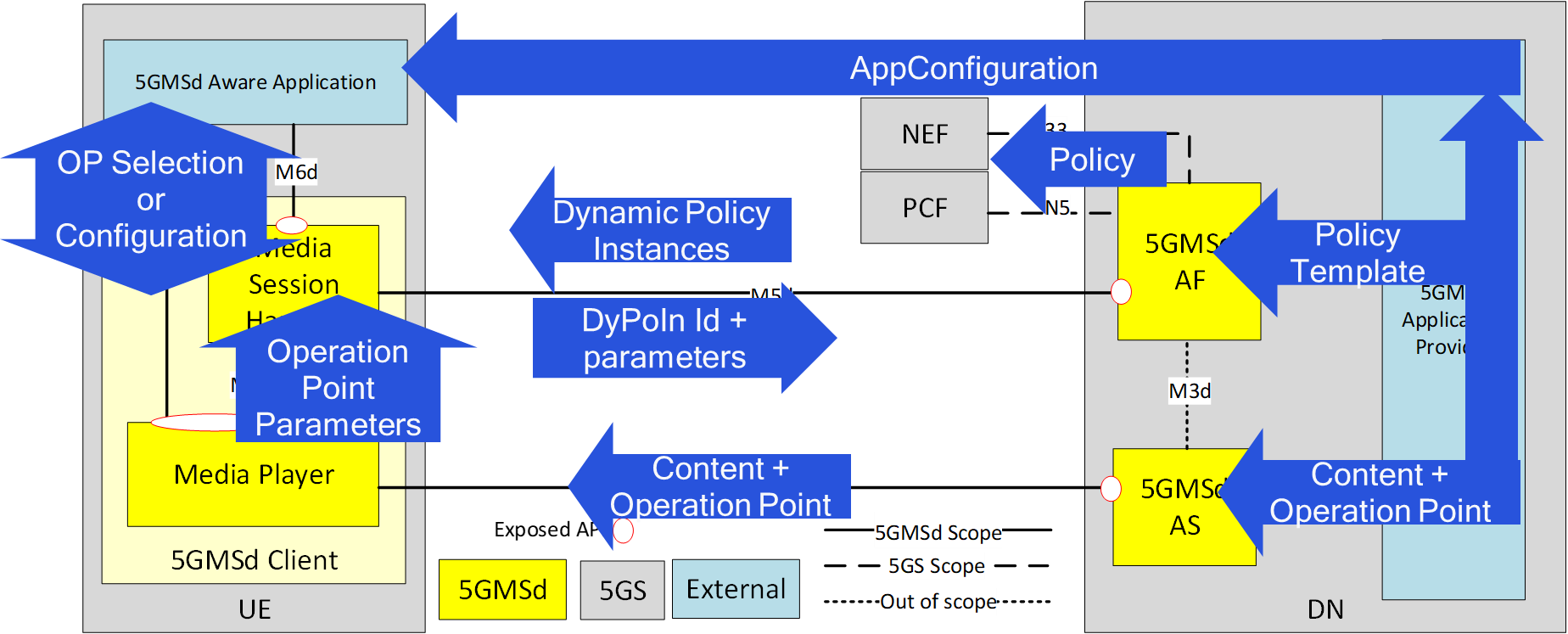


Figure 2 Operation Points in 5G Media Streaming

In particular, clause 12 of TS 26.512 defines details on how to obtain status information. Table 13.2.6-1 provides a list of dynamically changing status information that can be obtained from the client.

Table 13.2.6-1: Dynamic Status information

|  |  |  |  |
| --- | --- | --- | --- |
| Status | Type | Parameter | Definition |
| AverageThroughput | float | none | Current average throughput computed in the ABR logic in bit/s. |
| BufferLength | float | MediaType  "video", "audio" and "subtitle" | Current length of the buffer for a given media type, in seconds. If no type is passed in, then the minimum of video, audio and subtitle buffer length is returned. NaN is returned if an invalid type is requested, the presentation does not contain that type, or if no arguments are passed and the presentation does not include any adaption sets of valid media type. |
| liveLatency | float | none | Current live stream latency in seconds based on the latency measurement. |
| MediaSetting[] | MPDAdaptationSet | MediaType  "video", "audio" and "subtitle" | Current media settings for each media type based on the CMAF Header and the MPD information based on the selected Adaptation Set for this media type. |
| MediaTime | float | None | Current media playback time from media playback platform. The media time is in seconds and is relative to the start of the playback and provides the media that is actually rendered. |
| PlaybackRate | float | None | The current rate of playback. For a video that is playing twice as fast as the default playback, the playbackRate value should be 2.00. |
| availableServiceDescriptions[] | Provides the available service descriptions |  | Provides the list of available selectable service descriptions with an id to select from. Those are either configured ones or the ones in the MPD. |
| availableMediaOptions[] | List of Adaptation Set or Preselection ids | MediaType  "video", "audio" "subtitle" "all" | Provides the list of available media options that can be selected by the application based on the capability discovery and the subset information. |
| Metrics[][] | Metrics |  | A data blob of metrics for each defined metrics collecting scheme. |

### Live services at different scale

Live TV services of different scale (professional, user-generated, session-based, etc.) are increasingly distributed over broadband and mobile networks. Live TV services are characterized by:

- scalability (in terms of concurrent users),

- consistent quality,

- high bandwidth requirements, and

- target latency constraints.

### Scalability

Consistent support of the distribution of such services to a different scale of users and in a concurrent fashion is a prime concern. 5G Media Streaming is expected to support such service distribution and end-to-end optimizations. Improvements and optimizations on the architectural level and stage 3 are expected to be studied.

### Consistent quality

TV Services are expected to provide a consistent quality over time. TS 26.512 [X] defines Operation Point parameters in Table 13.2.6-2, repeated below, to which the client is configured. This configuration setting may be included in the manifest or may be provided through application means. Consistent quality can be defined that the service stays within the operation point boundaries to the largest extent. Specific aspects are:

* Meeting the latency requirements, namely staying at the target, not exceeding the maximum, and not falling below the minimum. Measuring deviation from the target, as well creating events when exceeding the boundaries, is relevant. For more details see clause 5.11.15.
* Meeting the playback rate, i.e. how often the playback rate is changed from 1.0, and if there are cases when the playback rate is outside of the range. For more details see clause 5.11.15.
* Staying within the boundaries of the bitrates and meeting the target is relevant as well. This measurement includes those cases for which the Bandwidth would have fallen to 0, i.e. the service is stalled. All of this can be measured by the client.

Generally, consistent quality refers to meet the below operation point parameters as shown in Table 5.11.1.3-1 (see also TS 26.512, Table 13.2.6-2). The Media Player and the network are expected to collaborate to meet the quality requirements.

Table 5.11.1.3-1: Operation Point Information (see TS 26.512, Table 13.2.6-2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OperationPoint | | | Operation Point Parameters | The currently configured operation point parameters according to which the DASH client is operating. |
|  | mode | | Enum | The following operation modes are defined:  live: The DASH client operates to maintain configured target latencies using playback rate adjustments and possibly resync.  vod: The DASH client operates without latency requirements and rebuffering may result in additional latencies |
|  | maxBufferTime | | Integer | maximum buffer time in milliseconds for the service. |
|  | switchBufferTime | | Integer | buffer time threshold below which the DASH clients attempts to switch Representations. |
|  | Latency | |  | Defines the latency parameters used by the DASH client when operating in live mode. |
|  |  | target | Integer | The target latency for the service in milliseconds. |
|  |  | max | Integer | The maximum latency for the service in milliseconds. |
|  |  | min | Integer | The maximum latency for the service in milliseconds. |
|  | PlaybackRate | | MediaType  audio, video, all | Defines the playback rate parameters used by the DASH client for catchup mode and deceleration to avoid buffer underruns and maintaining target latencies. |
|  |  | max | Real | The maximum playback rate for the purposes of automatically adjusting playback latency and buffer occupancy during normal playback, where 1.0 is normal playback speed. |
|  |  | min | Real | The minimum playback rate for the purposes of automatically adjusting playback latency and buffer occupancy during normal playback, where 1.0 is normal playback speed. |
|  | Bandwidth | |  | Defines the operating bandwidth parameters used by the DASH client used for a specific media type or aggregated. The values are on IP level. |
|  |  | target | Integer | The target bandwidth for the service in bit/s that the client is configured to consume. |
|  |  | max | Integer | The maximum bandwidth for the service in bit/s that the client is configured to consume. |
|  |  | min | Integer | The minimum bandwidth for the service in bit/s that the client is configured to consume. |
|  | PlayerSpecificParameters | |  | Player specific parameters may be provided, for example about the used algorithm, etc. |

### Deployment Architectures

A deployment architecture suitable for low-latency CMAF streaming is shown in Figure 5.11.2.1-1.

Timeline

Description automatically generated

Figure 5.11.2.1-1 Deployment architecture for low-latency CMAF streaming

In this case:

1. A live stream is ingested into a live encoder.

2. The encoded stream is packaged into CMAF chunks.

3. The packaged CMAF chunks are uploaded to an origin server using chunked transfer encoding input.

4. Segments are then available for retrieval by a CDN on demand and moved through the CDN all the way to the client.

### Operation Point – Establishment and Monitoring

This clause deals with providing consistent quality as part of an operation point. Figure 5.11.2.2-1 provides a basic setup on how operation points and policies can be matched. The content defined Operation points as shown in the user plane setup in Figure 5.11.2.2-1. Service Operation points define long lived profiles that will be used by streaming sessions as references. Based on communication with the application, the device characteristics, and so on, the media player selects an operation point that is determined by parameters as defined in Table 5.11.1.3-1. Based on these parameters, the policies in the 5G network are established, based on well defined policy templates. Policy templates represent long term agreements made between the AP and the MNO. The streaming session uses at most one of the allowed policy templates at any point in time.

Diagram

Description automatically generated

Figure 5.11.2.2-1 Operation Point work flow

### Key Issues for DASH

In order to make full use of the above functionality, the details on operation points can be carried in the application space. However, for better interoperability, a significant benefit is the addition on Operation Points associated to content in the MPD. The Service Description in DASH is exactly built for this.

We believe we should solicit input from Service Providers on static and dynamic service parameters that can be mapped to delivery optimizations.

We also believe that different operation points need to be mapped to content options, for example

* HD content (define the associated media and the required operation points)
* FullHD content (define the associated media and the required operation points)

### Proposed Updates Service Description

K3.X Operating Mode

Table K.X defines the service description parameters for operation. The keys in Table K.X shall be used to refer to the operating bandwidth as defined in Table K.X.

Table K.X — Operating Mode

|  |  |  |
| --- | --- | --- |
| **Key** | **Type** | **Description** |
| Id | Integer | An identifier for the operation mode |
| Mode | Enum | The following operation modes are defined:   * live: The DASH client operates to maintain configured target latencies using playback rate adjustments and possibly resync. * vod: The DASH client operates without latency requirements and rebuffering may result in additional latencies |
| maxBufferTime | Integer | maximum buffer time in milliseconds for the service. |
| switchBufferTime | Integer | buffer time threshold below which the DASH client is recommended to switch Representations. |
| Subset | Integer List | Provides the list of subsets in the MPD to which this Operating mode applies. |

## Extending the Service Description (m67854)

### Introduction

This document discusses extensions to Service Description to permit dynamic configurations of the network and to provide additional configuration options for clients.

This document discusses extensions to Service Description to permit dynamic configurations of the network and to provide additional configuration options for clients.

### Service Description Today

In the MPEG DASH, in Annex K the service description has been defined that allows to configure the DASH client for playback and other operational purposes. The service description is formalizing the API description. The configuration of the API is supported with typical API communication, i.e. also feedback from engine may be provided as well as status reporting/notifications. In case no app is available, MPEG-DASH permits to send configuration information in the MPD on MPD and Period level. This allows for static or semi-static updates of the service description.



Today the following parameters can be set in the Service Description:

* Service Latency
* Playback Rate
* Operating Quality
* Operating Bandwidth
* Content Steering
* Client-Metadata Reporting Configuration

### DASH.js Settings

DASH.js allows the following settings according to https://cdn.dashjs.org/latest/jsdoc/module-Settings.html

* [AbandonRequestRuleParameters](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~AbandonRequestRuleParameters)
* [AbrRulesParameters](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~AbrRulesParameters)
* [AbrSettings](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~AbrSettings)
* [AudioVideoSettings](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~AudioVideoSettings)
* [Buffer](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~Buffer)
* [CachingInfoSettings](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~CachingInfoSettings)
* [Capabilities](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~Capabilities)
* [CmcdSettings](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~CmcdSettings)
* [CmsdAbrSettings](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~CmsdAbrSettings)
* [CmsdSettings](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~CmsdSettings)
* [DebugSettings](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~DebugSettings)
* [ErrorSettings](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~ErrorSettings)
* [Gaps](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~Gaps)
* [LiveCatchupSettings](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~LiveCatchupSettings) 🡺 Playback Rate
* [LiveDelay](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~LiveDelay)
* [Metrics](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~Metrics)
* [PlayerSettings](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~PlayerSettings)
* [Protection](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~Protection)
* [RequestTypeSettings](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~RequestTypeSettings)
* [Scheduling](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~Scheduling)
* [StreamingSettings](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~StreamingSettings)
* [Text](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~Text)
* [TimeShiftBuffer](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~TimeShiftBuffer)
* [UtcSynchronizationSettings](https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~UtcSynchronizationSettings)

Note that a subset of the above settings can be configured through service description.

DASH.js also permits to provide notifications from the player to the application, for example here: https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html

### Issues

The following three issues are observed:

1. Service description in MPEG-DASH only supports a subset of the configurations of dash.js. While several may not be relevant to be configured, a few of them are for sure. A careful checking should be done, which dash.js settings should be permitted to be configured from remote.
2. The service configuration can be sent, but if the configuration is not successful or if other type of status or notifications are generated in the DASH client, this information cannot be provided as part of the MPD and the over the network. Configurations are supported by feedback, notifications and events. However, in the absences of an app, such events are not fired and not made available to the sender. For example, CMCD does not allow to send dash.js events and notifications.

### Solution for 1: Extend Service Description with additional parameters

It is proposed to review the dash.js settings carefully. Based on the initial review, it is proposed to add to the service description the following settings:

1. Recommended ABR strategy: <https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~AbrSettings> – this allows the service provider to configure the ABR algorithm
2. Buffer Configuration: <https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~Buffer> – this allows the service provider to configure buffer handling.
3. Gaps: <https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~Gaps> this allows to support consistent gap handling
4. Scheduling: https://cdn.dashjs.org/latest/jsdoc/module-Settings.html#~Scheduling

### Solution for 2: Networking Notifications

#### Overview

Notifications of the DASH client may be of relevance not only for the application, but also for the network. However, not all notifications may be of the same relevance. Some of the events and notifications may also be included in some reporting schemes. The events highlighted in **bold and red** below may be suitable for being networked

1. [AST\_IN\_FUTURE](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:AST_IN_FUTURE)
2. [BASE\_URLS\_UPDATED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:BASE_URLS_UPDATED)
3. [BUFFER\_EMPTY](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:BUFFER_EMPTY)
4. [BUFFER\_LEVEL\_STATE\_CHANGED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:BUFFER_LEVEL_STATE_CHANGED)
5. [BUFFER\_LEVEL\_UPDATED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:BUFFER_LEVEL_UPDATED)
6. [BUFFER\_LOADED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:BUFFER_LOADED)
7. [CAN\_PLAY](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:CAN_PLAY)
8. [CAN\_PLAY\_THROUGH](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:CAN_PLAY_THROUGH)
9. [CAPTION\_CONTAINER\_RESIZE](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:CAPTION_CONTAINER_RESIZE)
10. [CAPTION\_RENDERED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:CAPTION_RENDERED)
11. [CONFORMANCE\_VIOLATION](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:CONFORMANCE_VIOLATION)
12. [CONTENT\_STEERING\_REQUEST\_COMPLETED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:CONTENT_STEERING_REQUEST_COMPLETED)
13. [CUE\_ENTER](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:CUE_ENTER)
14. [CUE\_ENTER](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:CUE_ENTER)
15. [DVB\_FONT\_DOWNLOAD\_ADDED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:DVB_FONT_DOWNLOAD_ADDED)
16. [DVB\_FONT\_DOWNLOAD\_COMPLETE](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:DVB_FONT_DOWNLOAD_COMPLETE)
17. [DVB\_FONT\_DOWNLOAD\_FAILED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:DVB_FONT_DOWNLOAD_FAILED)
18. [DYNAMIC\_TO\_STATIC](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:DYNAMIC_TO_STATIC)
19. [ERROR](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:ERROR)
20. [EVENT\_MODE\_ON\_RECEIVE](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:EVENT_MODE_ON_RECEIVE)
21. [EVENT\_MODE\_ON\_START](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:EVENT_MODE_ON_START)
22. [FRAGMENT\_LOADING\_ABANDONED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:FRAGMENT_LOADING_ABANDONED)
23. [FRAGMENT\_LOADING\_COMPLETED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:FRAGMENT_LOADING_COMPLETED)
24. [FRAGMENT\_LOADING\_PROGRESS](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:FRAGMENT_LOADING_PROGRESS)
25. [FRAGMENT\_LOADING\_STARTED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:FRAGMENT_LOADING_STARTED)
26. [INBAND\_PRFT](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:INBAND_PRFT)
27. [LOG](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:LOG)
28. [MANIFEST\_LOADED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:MANIFEST_LOADED)
29. [MANIFEST\_LOADING\_FINISHED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:MANIFEST_LOADING_FINISHED)
30. [MANIFEST\_LOADING\_STARTED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:MANIFEST_LOADING_STARTED)
31. [MANIFEST\_VALIDITY\_CHANGED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:MANIFEST_VALIDITY_CHANGED)
32. [METRIC\_ADDED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:METRIC_ADDED)
33. [METRIC\_CHANGED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:METRIC_CHANGED)
34. [METRIC\_UPDATED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:METRIC_UPDATED)
35. [METRICS\_CHANGED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:METRICS_CHANGED)
36. [PERIOD\_SWITCH\_COMPLETED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PERIOD_SWITCH_COMPLETED)
37. [PERIOD\_SWITCH\_STARTED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PERIOD_SWITCH_STARTED)
38. [PLAYBACK\_ENDED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_ENDED)
39. [PLAYBACK\_ERROR](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_ERROR)
40. [PLAYBACK\_LOADED\_DATA](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_LOADED_DATA)
41. [PLAYBACK\_METADATA\_LOADED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_METADATA_LOADED)
42. [PLAYBACK\_NOT\_ALLOWED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_NOT_ALLOWED)
43. [PLAYBACK\_PAUSED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_PAUSED)
44. [PLAYBACK\_PLAYING](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_PLAYING)
45. [PLAYBACK\_PROGRESS](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_PROGRESS)
46. [PLAYBACK\_RATE\_CHANGED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_RATE_CHANGED)
47. [PLAYBACK\_SEEKED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_SEEKED)
48. [PLAYBACK\_SEEKING](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_SEEKING)
49. [PLAYBACK\_STALLED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_STALLED)
50. [PLAYBACK\_STARTED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_STARTED)
51. [PLAYBACK\_TIME\_UPDATED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_TIME_UPDATED)
52. [PLAYBACK\_VOLUME\_CHANGED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_VOLUME_CHANGED)
53. [PLAYBACK\_WAITING](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:PLAYBACK_WAITING)
54. [QUALITY\_CHANGE\_RENDERED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:QUALITY_CHANGE_RENDERED)
55. [QUALITY\_CHANGE\_REQUESTED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:QUALITY_CHANGE_REQUESTED)
56. [REPRESENTATION\_SWITCH](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:REPRESENTATION_SWITCH)
57. [STREAM\_ACTIVATED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:STREAM_ACTIVATED)
58. [STREAM\_DEACTIVATED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:STREAM_DEACTIVATED)
59. [STREAM\_INITIALIZED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:STREAM_INITIALIZED)
60. [STREAM\_INITIALIZING](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:STREAM_INITIALIZING)
61. [STREAM\_TEARDOWN\_COMPLETE](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:STREAM_TEARDOWN_COMPLETE)
62. [STREAM\_UPDATED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:STREAM_UPDATED)
63. [TEXT\_TRACK\_ADDED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:TEXT_TRACK_ADDED)
64. [TEXT\_TRACKS\_ADDED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:TEXT_TRACKS_ADDED)
65. [THROUGHPUT\_MEASUREMENT\_STORED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:THROUGHPUT_MEASUREMENT_STORED)
66. [TRACK\_CHANGE\_RENDERED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:TRACK_CHANGE_RENDERED)
67. [TTML\_PARSED](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:TTML_PARSED)
68. [TTML\_TO\_PARSE](https://cdn.dashjs.org/latest/jsdoc/MediaPlayerEvents.html#event:TTML_TO_PARSE)

#### Options for sending Events

Two options are considered for sending such notifications:

1. As part of CMCD (and actually some are already supported)
   1. This would require a careful analysis of the data above
   2. Relevant notification events not yet included in CMCD may be added
2. As part of a callback event that is initiated through a service description extension
   1. In this case, a callback is triggered whenever an event is happening.
   2. This can be combined with CMCD and the reporting of specific CMCD metrics is only be done as part of a call back

# MSE implementation of inband events (m56684)

Currently there are activities in CTA common media library to create the polyfill library to handle inband events.

# Clarifying pre-roll signaling for seamless content splicing across MPEG-DASH Periods (m56890)

Note: the issue of audio prerolling during period transitioning is being discussed in ISOBMFF as well as CMAF. We will investigate how it should be the best to address**.**

## Introduction

In our experiments on splicing, DASH-ing, and playing content back we have encountered issues to provide the required information to the playback device to provide a perfect experience (from a Systems perspective). These issues may happen with audio, video, and other media as well.

Splicing points may be mandated whenever the most suitable from a system-level perspective i.e. at segment boundaries or when SAP types 1 or 2 occur. However, this is not always possible or desirable (e.g. because it would require to re-process the content or e.g. because for historical reasons SAP Types are not well signaled).

For these use-cases we realized that MPEG-DASH (and other HTTP Adaptive Streaming (HAS) technologies) is missing signaling to allow a player to get some context information that is needed to reconstruct the container-advertised content.

## Related contributions

Contributions m55420 (October 2020) and m55478 (liaison letter from DVB) raise questions around the correct usage of DASH multi-period with regards with content continuity. Both contributions raise pragmatic questions. The present contribution aims at apprehending the same category of issues from a more generic perspective.

## Discussion

### Pre-roll context

Issues when splicing occur when the splice point doesn’t respect the SAP Type 1 or 2 constraints. Then the effective presentation duration may differ from the presentation duration advertised at the container and MPD levels because the player (in particular an elementary stream decoder) doesn’t have the decoding context which would allow to fully decode the segment, ensuring a perfect playback experience.

The difference between the decoded data duration and the presentation duration is usually handled using edit lists. In this case the edit list entries serve both to compensate some introduced delays as well as adjusting the duration (hence discarding data that is not intended to be presented).

Usually, a single decoder is used to decode a single content. Hence a single initialization segment can contain some general information about the processing of this non-presented data.

As SAP Types postdates the introduction of encoding technologies using pre-roll some streams (typically audio ones) are still commonly signaled as SAP Type 1 while this is not correct.

The pre-roll information should be present inside the container in conformance to the applicable standards.

## Use-cases

### Ad insertion using multiple Periods

In this use case, advertisement content (Period #2) is inserted in the middle of a main content (Periods #1 and #3). When switching back from the ad to the main content some players may need to reinitialize themselves. In this case the pre-roll information and associated data shall be present to ensure a perfect playback experience.

In current HAS deployments, this pre-roll data is located in a previous segment whose URL is unknown at the HAS layer, as it requires identifying the pre-roll count of the first sample of the new segment to play.

### Open-GOPs splicing

Open-GOPs have been largely avoided by the industry due to the lack of proper description of how to handle them. As codecs using similar mechanisms become more widespread, the information about how to reconstruct a full signal over the advertised period is more necessary.

## Discussions

There are two possible approaches to solve this issue:

* Identify pre-roll data as part of the HAS manifest.
* Embed pre-roll data in media segments.

Working at the manifest level has the following benefits:

* No changes required to media segments.

Working at the manifest level has the following drawbacks:

* Pre-roll URL needs to be indicated for each switching point, i.e. potentially for each segment.
* The system is dependent of the length of the time-shift buffer and requires the past segment containing the pre-roll content to be available; this is usually not the case in broadcast environments such as ROUTE.
* The solution will have to be declined for each possible manifest formats.
* The URL might not be fine-grained enough, resulting in fetching more data than needed (potentially a full segment or subsegment), i.e. the moof+mdat identified is much larger than the pre-roll data. Reducing the duration of subsegments to match the pre-roll duration is possible but will increase the overhead (for AAC, this could mean one moof+mdat per AAC frame).
* It requires the initialization segment (in the case of ISOBMFF) to be the same between the pre-roll segment and the actual segment, or a fetch of the previous init segment to parse the pre-roll data.

Working at the segment level has the following benefits:

* Independent from the HAS solution (DASH, HLS, pure CMAF).
* Only the required pre-roll data is copied in the media segment, lowering overhead when switching.

Working at the segment level has the following drawbacks:

* It requires update to the media segment parsing.
* It may require updates of the media pipeline, or an adaptation layer reformatting the segment at switch point.
* The solution will have to be declined for each possible segment formats
* Duplication of media data.

A third approach, specific to MPEG-DASH, could be to use the BitstreamSwitching segment, currently not defined for ISOBMFF. Such a bitstream switching segment would only contain the pre-roll AUs, formatted according to the current init segment. It will still result in media duplication, but does not require any modification to the media segment.

A fourth approach, only valid for splice-out case (main content resume), would be to not use period continuity signaling and copy over the pre-roll data in the first segment of the new period.

## Conclusion

We believe it is MPEG responsibility to address these issues urgently, and kindly request WG3 to investigate possible actions to define guidelines or technical proposals towards a unified, decoding-exact handling of splicing and non-SAP1 switching.

# Enabling CMCD beaconing (m65127)

<https://mpeg.expert/software/MPEG/Systems/DASH/spec/-/issues/402>

Note: This text replaces Callback event version 2 (m64319) <https://mpeg.expert/software/MPEG/Systems/DASH/spec/-/issues/379>.

Note: CMCD V2 supports beaconing but in order to add the support in DASH, we are waiting for the publication of CMCD V2 spec.

## Introduction

Observability is important for video streaming systems because it allows service providers to monitor the health of the streaming platform, identify and diagnose issues that impact user experience, and optimize the performance of the system to ensure a high-quality streaming experience. The streaming player behavior is what ultimately determines the viewers’ QoE, hence observing the player at very low level in real time can provide extremely valuable data that can be further used for data-driven QoE improvement.

Video streaming is a complex and dynamic system that involves multiple components, including the client device, the network infrastructure, the content delivery network (CDN), and the video player. Letting the streaming player have insights on the state of the whole system may let it make data-driven decisions such as estimating the optimal bitrates, optimal CDNs, and HTTP GET request times.

CTA CMCD (CTA-5004) implementations use URL query parameters and headers of HTTP GET requests for segments and MPDs to communicate information regarding streaming sessions being played out. While this works remarkably well, the mode of operation where a periodic HTTP POST to a provider endpoint as opposed to “piggybacking” on other requests is missing.

The contribution accepted at the Geneva meeting into AMD2 contained an option of JSON output (defined by CTA for the purpose), however there was no way of defininig the URL to which this JSON would be posted. Two options for achieving the above were previously suggested: (a) introducing a new event type with recurrence expressed in the event body, and (b) using a URL specified directly in the CMCD descriptor.

This contribution proposes to restore the JSON option of CMCD using the approach specified in (a), for simplicity purposes. The proposal modifies the **ClientDataReporting** element, in order to generalize beaconing for reporting purposes:

| **Element or Attribute Name** | | **Use** | **Description** |
| --- | --- | --- | --- |
| **ClientDataReporting** | |  | An element that provides information about client data reporting as defined in subclause K.3.7. |
|  | @serviceLocations | O | See serviceLocations in Table K.3.7-1. |
|  | @adaptationSets | O | See adaptationSets in Table K.3.7-1. |
|  | @beaconingURL | CM | URL to which reporting should be done. If absent, reporting is performed using other request types (e.g. segments or MPD)  If the CMCDParameters element is present, reporting shall be done using an HTTP POST method with body of the request being JSON-formatted CMCD, per CTA 5004.  This attribute shall be present if and only if the **CMCDParameters**@mode is set to “json” |
|  | @frequency | CM | Frequency (in seconds, possibly fractional) at which beconing requests shall be issued. Present if and only if the @beaconingURL is present. |
|  | **CMCDParameters** | 0 … 1 | Defines reporting system parameters to send back client data for the above Service Locations and and Adaptation Sets for CMCD. For details refer to clause K.4.2.7.2. |
| **Legend:**  For attributes: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory, F=fixed.  For elements: <minOccurs>...<maxOccurs> (N=unbounded)  The conditions only hold without using xlink:href. If linking is used, then all attributes are "optional" and <minOccurs=0>  Elements are **bold**; attributes are non-bold and preceded with an @, List of elements and attributes is in ***italics bold*** referring to those taken from the Base type that has been extended by this type. | | | |

We further remove the prohibition on the json format.

**Table K.3.7-2 — CMCD specific parameters**

|  |  |  |
| --- | --- | --- |
| **Key** | **Type** | **Description** |
| Version | unsigned int | specifies the highest *CMCD version* as defined in CTA-5004 that is accepted by the reporting server.  If absent, the version is assumed to be version 1 as defined in CTA-5004. |
| mode | string | specifies the data transition mode how the media client shall send the media client data as defined in clause 2 of CTA-5004.  The permitted options are "query", "header", and “json”. "header" refers to the mode defined in clause 2.1 of CTA-5004. "query" refers to the mode defined in clause 2.2 of CTA-5004.  If the “json” method is used, @beconingURL and @frequency attributes shall be set.  If the value is absent, the "query" method shall be used.  ~~Note: the third method, including the data in a JSON object, is not defined in this standard as CTA-5004 does not define a detailed protocol.~~ |

## Notes from MPEG#144

1. CMCD currently does not include any reporting protocol
2. In past, we didn't define any reporting mechanism for the DASH metric
3. This should be provided as part of API
4. Recommendation:
   1. Update the contribution to address the API (not implemented in above yet)
   2. Work with CTA WAVE on the scope of work in CMCD V2 and whether it will cover any protocol aspect.

Editor’s note: Recommend a solution to be considered for ISOBMFF before we do anything in DASH.

# Signaling maximum segment size (m65860)

[MPEG/Systems/DASH/spec#422MPEG/Systems/DASH/spec#422](https://git.mpeg.expert/MPEG/Systems/DASH/spec/-/issues/422)

## Introduction

The DASH **Representation**@bandwidth model is defined as a rate sufficient to transmit **MPD**@minBufferTime. Unfortunately, it does not explicitly specify either units or mode of operation, similarly to what has been done by the MPEG video standards for VBV and HRD. As a result, in many cases the MPD contains a bitrate equivalent to the maximum bitrate as stated in the HRD. The problem with the approach is that for shorter segments (e.g. 2s) and longer CPBs (e.g. 2s when converted into time units) the segment size often exceeds the expected maximum size of a segment as calculated given the value of the @bandwidth attribute. Convergence to the @bandwidth value is only guaranteed with segments longer than 2\*CPB. This is problematic, especially for low-latency applications, as the client cannot always correctly predict the worst case time it will take it to download a segment.

The above problem is only relevant for live profile services – in case of on demand profile `sidx` provides precise information making the question moot.

Apple HLS definition is entirely different and operates in units of segments. The bandwidth signaled is defined as the maximum bitrate for a sequence of segments between 0.5x and 1.5x of a target segment duration. They further recommend not to exceed the stated rate by more than 10%. This definition is not as elegant but provides the player with hard guarantees regarding the size of the segment. Having an HLS-compatible definition will make the HLS-to-DASH and DASH-to-HLS translation simpler.

The above proposal was discussed during MPEG #143 and added to the AhG mandate and to the TuC. The discussion at <https://mpeg.expert/software/MPEG/Systems/DASH/spec/-/issues/368> posed two questions:

1. What is the operational model?
2. How does this work with SegmentTimeline when there is no nominal duration?

## Proposal

We propose to define a concept of *segment bandwidth*, conceptually defined as an peak bitrate of **MPD**@maximumSegmentDuration worth of consecutive media segments (rounded up to the nearest segment boundary).

### Definition

Let segment buffer *SBi* be comprised of the shortest sequence of one or more consecutive segments or segment sequences *S(i)..S(i+k)* with combined media duration between 0.5×**MPD**@maximumSegmentDuration and 1.5×**MPD**@maximumSegmentDuration. *Instantaneous segment bandwidth* is defined as the sum of the sizes of the above segments (or segment sequences) in bytes divided by their combined duration.

NOTE: the above variation is needed for avoiding very short video segments which inherently have higher rate (due to higher proportion of much larger I frames to P/B frames) and skew the rate calculation.

We further define a **Representation**@segmentBandwidth attribute as the maximum instantaneous bitrate of available segments.

### Operational model

#### Client operation

The client can make the following assumptions regarding the segment duration *SDe* prior to reading it:

1. The segment duration may not exceed **MPD**@maximumSegmentDuration *(MSD).* Thus *SDe = MSD*
2. The segment duration is within tolerance *T* from **SegmentTimeline**@duration *D*. Here tolerance is defined as *T =* *min(0.5*×*D, (1+***SegmentTimeline**@tolerance*)*×*D)* and *SDe =* *min(MSD, D + T)*
3. The segment duration is specified in the **S**@d attribute, thus *SDe* = **S**@d
4. The subsegment duration is specified in `sidx` along with size, hence the question is moot (segment size is precisely known)

Given the above assumptions, the maximum segment size in bytes *MSS* anticipated by the client is *SDe*×**Representation**@segmentBandwidth

#### Compliance

The segment shall never exceed the MSS.

For live, MSS is a bound defined by the encoder based on its expectation of the maximum size.

Note that HLS compliance allows an “overshoot” of 10% over the segment bandwidth stated in the BANDWIDTH attribute of the #EXT-X-STREAM-INF tag. This makes the relationship somewhat awkward as MSS is always 110% of the value of BANDWIDTH. It may be better to allow the same 10% overshoot to make conversion simpler (i.e., we have the same value in both DASH MPD and HLS playlist).

# Extending copyright license signaling (m66512)

<https://git.mpeg.expert/MPEG/Systems/DASH/spec/-/issues/431>

## Introduction

This contribution proposes extending the copyright license signaling in the DASH standard.

## Background

The MPD has an element, ProgramInformation, at the MPD level. This element contains the copyright information for the entire MPD. This element can be used once or multiple times. The common use of multiple ProgramInformation is to describe the copyright information in multiple languages.

Table 31 — Program information semantics

| **Element or Attribute Name** | | | **Use** | **Description** |
| --- | --- | --- | --- | --- |
|  | ProgramInformation | |  | specifies descriptive information about the program |
|  |  | @lang | O | Declares the language code(s) for this Program Information. The syntax and semantics according to IETF RFC 5646 shall be applied.  If not present, the value is unknown. |
|  |  | @moreInformationURL | O | If provided, this attribute specifies an absolute URL which provides more information about the Media Presentation.  If not present, the value is unknown. |
|  |  | Title | 0 ... 1 | specifies the title for the Media Presentation. |
|  |  | Source | 0 ... 1 | specifies information about the original source (for example content provider) of the Media Presentation. |
|  |  | Copyright | 0 ... 1 | specifies a copyright statement for the Media Presentation, usually starting with the copyright symbol, unicode U+00A9. |
| **Key**  For attributes: M=mandatory, O=optional, OD=optional with default Value, CM=conditionally mandatory  For elements: <minOccurs>...<maxOccurs> (N=unbounded)  Elements are bold; attributes are non-bold and preceded with an @. | | | | |

<xs:complexType name="ProgramInformationType">

<xs:annotation>

<xs:documentation xml:lang="en">

**Program Information**

</xs:documentation>

</xs:annotation>

<xs:sequence>

<xs:element name="Title" type="xs:string" minOccurs="0"/>

<xs:element name="Source" type="xs:string" minOccurs="0"/>

<xs:element name="Copyright" type="xs:string" minOccurs="0"/>

<xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>

</xs:sequence>

<xs:attribute name="lang" type="xs:language"/>

<xs:attribute name="moreInformationURL" type="xs:anyURI"/>

<xs:anyAttribute namespace="##other" processContents="lax"/>

</xs:complexType>

## Use case and justification

While the DASH standard provides ProgramInformation to describe copyright information for the entire program, it lacks signaling different copyright information per period. A simple use case is that the copyright license of the content may change per period since the program may consist of main content and ads or a splice of different content.

Furthermore, signaling the copyright license is becoming more important in light of the recent legislative activities on AI deep fake content. Contribution m66513 describes two legislative activities for signaling the marking of AI-generated/altered content. At the same time, there have also been significant activities on copyright laws and the applicability of those laws to the content that is altered or generated by AI in major jurisdictions (the United States, the European Union, the United Kingdom, and China). Regardless of the results of this discussion, it seems that if a part of the content is altered, the copyright license for that part might be different than the original content copyright license. Therefore, providing the means of providing clear copyright licenses for different parts of content is needed.

## Proposed solution

We define an element for copyright information:

| **Element or Attribute Name** | | | **Use** | **Description** |
| --- | --- | --- | --- | --- |
|  | CopyrightInfo | |  | Specifies copyright information |
|  |  | @lang | O | Declares the language code(s) for this Program Information. The syntax and semantics according to IETF RFC 5646 shall be applied.  If not present, the value is unknown. |
|  |  | @moreInformationURL | O | If provided, this attribute specifies an absolute URL which provides more information about the Media Presentation.  If not present, the value is unknown. |
|  |  | @owner | O | indicating the owner of the content. |
|  |  | @year | O | The year(s) of the copyright. |
|  |  | @allRightsReserved | O | If true, no right is granted by the copyright owner. |
|  |  | @license | O | Describes the licensing information. |
| **Key**  For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory  For elements: <minOccurs>...<maxOccurs> (N=unbounded)  Elements are bold; attributes are non-bold and preceded with an @. | | | | |

We add the above element to MPD as part of ProgramInformation, Period, AdaptationSet, and ContentComponent elements. The semantics for ProgramInformation is shown below.

| **Element or Attribute Name** | | | **Use** | **Description** |
| --- | --- | --- | --- | --- |
|  | ProgramInformation | |  | specifies descriptive information about the program |
|  |  | @lang | O | Declares the language code(s) for this Program Information. The syntax and semantics according to IETF RFC 5646 shall be applied.  If not present, the value is unknown. |
|  |  | @moreInformationURL | O | If provided, this attribute specifies an absolute URL which provides more information about the Media Presentation.  If not present, the value is unknown. |
|  |  | Title | 0 ... 1 | specifies the title for the Media Presentation. |
|  |  | Source | 0 ... 1 | specifies information about the original source (for example content provider) of the Media Presentation. |
|  |  | Copyright | 0 ... 1 | specifies a copyright statement for the Media Presentation, usually starting with the copyright symbol, unicode U+00A9.  At most one Copyright or CopyrightInfo element at most shall be present. |
|  |  | CopyrightInfo | 0 ... 1 | specifies the extended copyright information. At most one Copyright or CopyrightInfo element at most shall be present. |
| **Key**  For attributes: M=mandatory, O=optional, OD=optional with default Value, CM=conditionally mandatory  For elements: <minOccurs>...<maxOccurs> (N=unbounded)  Elements are bold; attributes are non-bold and preceded with an @. | | | | |

Similarly, Period, the proposed CopyrightInfo element can be added to the AdaptationSet, and ContentComponent elements.

### Alternative solution

Alternatively, a supplemental descriptor can be used for signaling the copyright information using the above URI in its schemeURI. Then the value in the descriptor can consist of a white space-separated list of key=value pairs, where the keys are shown in the table below:

|  |  |  |
| --- | --- | --- |
| Key | **Use** | Description |
| DASHCRKEY-lang | M | Declares the language code(s) for this Program Information. The syntax and semantics according to IETF RFC 5646 shall be applied.  If not present, the value is unknown. |
| DASHCRKEY-moreInformationURL | O | If provided, this attribute specifies an absolute URL which provides more information about the Media Presentation.  If not present, the value is unknown. |
| DASHCRKEY-owner | O | indicating the owner of the content. |
| DASHCRKEY-year | O | The year(s) of the copyright. |
| DASHCRKEY-allRightsReserved | O | If true, no right is granted by the copyright owner. |
| DASHCRKEY-license | O | Describes the licensing information. |

The prefix “DASHCPKEY-“ in each key is used to identify a key from strings used in the values. Therefore, the use of this prefix is not allowed in any values.

One or more descriptors can exist in an element.

The above descriptor can only be used at the MPD, Period, AdaptationSet, or ContentComponent elements.

## Notes from MPEG#145

1. We are not in the business of defining what information should be carried. that can be defined by other standards or consortia, such as MPEG smart contract, EIDR, ADID.
2. We can investigate how to carry such information at lower levels than MPD, at period, at adaptation set, subsets, etc.
3. We need a client processing model, what does the client do with this information?
4. One use case would be the copyright of derivative work/added content, such as sign language video.

DASH events for C2PA metadata signaling (m72350)

## Abstract

This clause proposes amending ISO/IEC 23009-1 to signal C2PA [1], or other authenticity, data through DASH events.

## Introduction

During the November 2024 SC 29/ WG 2 meeting, the draft requirements on technologies for multimedia authentication were approved as WG02 N00413. During the same meeting, input contribution m70452 argued for the use of C2PA to meet the requirements developed by WG 2. During the January 2025 WG 2 & 3 meetings, input contribution m71125 presented a proof of concept for the delivery of C2PA information using DASH events. That contribution listed how the described approach meets the requirements adopted by WG 2 for delivering provenance and authenticity information. WG 3 established an AhG to study the topic with one of it’s mandates being the identification of how provenance and authenticity information can be provided using existing standards (mandate 3) and another to solicit potential solutions to fill gaps in existing systems standards to meet the requirements for providing authenticity and provenance information (mandate 4). The current contribution describes how DASH can be amended to allow the interoperable delivery of provenance and authenticity information.

The following section is mostly repeated from m71125 as a reminder of the workflow being addressed in this contribution.

## C2PA based streaming PoC

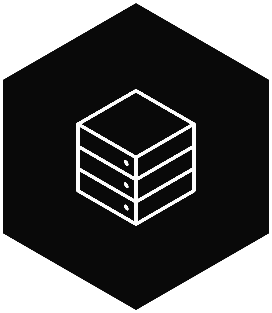
Figure 1 shows the system used for the PoC. A modified packager is used to create the manifest for each segment of media based on the packager’s configuration. The configuration includes:

1. The potential manipulation of the media (for example transcoding).
2. The signature algorithm to be used.
3. The signature key to be used (to be applied by the packager).
4. The structure of the C2PA manifest that must be created (the elements of the manifest to be completed).
5. Whether or not the ingested media carries C2PA information and it’s location (the location of the manifest or manifest store).

The modified packager has a C2PA engine (module) that:

1. Receives the media data (as segments).
2. Generates the required C2PA manifest data as either a single manifest or part of a store for the received media.

Finally, the modified packager may generate an in-band event stream that is compliant with ISO/IEC 23001-18 and ISO/IEC 23009-1 with the events added to the segments in each representation, or it may generate and out of band event stream. The message data of each event is the C2PA data associated with the media for media segment during the time period covered by the event (i.e. the media segment). The representation made available to the player includes this event stream so that a modified DASH player can validate the provenance information provided in the event stream. Each event has the same active time as the segment it is associated with.



C2PA engine

C2PA configuration interface

Media segments

C2PA manifests

Media presentation with C2PA event stream

DASH player with C2PA validation

VOD

Live

Figure 1 C2PA PoC system diagram

## Proposed modifications to ISO/IEC 23009-1

In order to enable the above described DASH based workflow, it is recommended that clause 5.10 in ISO/IEC 23009-1 be modified to include a new sub-clause (which would currently be 5.10.5) that describes media provenance and authenticity events. Both out of band and in band event streams should be specified.

For the case of signaling C2PA data, it is proposed that the scheme identifier urn:mpeg:dash:event:c2pa:21 be specified for use by either in-band or out of band event streams, indicating that the event carries C2PA data that is conformant with version 2.1 of the C2PA specification.

It is further proposed that the @id attribute for provenance and authenticity events shall be present and unique for each event. The @status attribute of the event, if present, shall not use the “update” value. The @status attribute should use the “repeat” value if the repeat content is being downloaded by the client.

The dispatching of these events shall be on-receive.

Table 1 defines the relevant Event and EventStream attributes for out of band events.

Table 1 C2PA specific Event and EventStream attributes for out of band events

|  |  |
| --- | --- |
| Key | Description |
| schemeIdURI  **EventStream**@schemeIdUri | Set to urn:mpeg:dash:event:c2pa:21 |
| presentation\_time  **Event**@presentationTime | Provides the media presentation time corresponding to the C2PA data. |
| duration  **Event**@duration | Shall be set to be the same as the duration of the segment to which the C2PA data is associated. |
| id  **Event**@id | Shall be present and unique per event in a presentation. |
| message  **Event** value | This shall be the C2PA manifest store or manifest data.  (Note that Event@messageData has been deprecated in the sixth edition in favour of using Event value, otherwise the Event@messageData would have been proposed for use in this case) |

For out of band C2PA events, the message shall contain a header indicating which representations the C2PA data is to be associated with as well as the C2PA manifest data as defined by the struct DASHC2PAEvent below:

aligned(8) struct DASHC2PAEvent

{

string representation\_id; // the representation to which this event belongs

string c2pa\_data; // c2pa manifest or manifest store

}

In the case of in band events the message attribute is not required and shall be ignored if present. The relevant C2PA data shall be found in the 'emsg'.message\_data[] field. Note that for in-band messages the association with the representation is clear.

## Recommendations

The proposed modifications listed above are recommended for inclusion in a possible amendment to ISO/IEC 23009-1.

## References

|  |  |
| --- | --- |
| [1] | Coalition for Content Provenance and Authenticity (C2PA), "Content Credentials : C2PA Technical Specification," [Online]. Available: https://c2pa.org/specifications/specifications/2.1/specs/C2PA\_Specification.html. |

Association of provenance and authenticity of multiple media representations in DASH manifest (m75658)

## Abstract

This contribution addresses the problem of associating the provenance and authenticity information of multiple representations (video, audio, text) that make up a presentation. The proposal is to insert a signature at the DASH MPD level. This signature is cryptographically associated with the content of the MPD, allowing for a chain of trust to be created that includes the association of representations as intended by the content creator.

## Introduction

Multimedia presentations composed from multiple media assets, such as a movie composed of video, audio and subtitles may be compromised by the disassociation of media assets from one another. For example, an audio track that was not intended by the content creators to be used along with a video track may be delivered in place of the intended audio track. There is also the possibility of audio and video being misaligned intentionally, however, that possibility is significantly reduced by the use of specifications such as C2PA (v2.3) [1] which specifies a mechanism for ensuring that segments in a track are being delivered in the intended order. This leaves the issue of associating whole component assets correctly with each other.

This proposal is for the inclusion of provenance and authenticity information as part of the DASH MPD.

## Proposal

The proposal is for the creator of the MPD that is accessed by a DASH client to include a signature element that includes a cryptographic hash of the other elements in the MPD.

It is proposed that a W3C <Signature> element [2] where the <SignedInfo> element references all the other elements of the MPD in which the signature is included. It is proposed that the signature element is included as the last element in the MPD.

By providing such a signature element the MPD creator is providing a signed, timestamped MPD that lists the assets included in a presentation as well as the time periods of the presentation. Any update of the MPD would include its own signature element. In this way, the client can validate that the media assets included in a presentation are associated with one another as intended by the creator of the MPD. The client can further log the received signatures as well as the validation results.

Assuming that the media assets have provenance and authenticity information embedded, the client is able to validate both the received media’s provenance and authenticity information as well as the intended association of the assets.

# Latency Reporting (m72386)

## Introduction

In the BBC Project Timbre work it was outlined that the latency of each audio segment was a key indicator of quality of service for the listener. The information may be exposed as an API in the network.

<https://www.bbc.co.uk/rd/blog/2024-03-project-timbre-investigating-mobile-coverage-for-live-radio-streaming-on-bbc-sounds>

In an Adaptive Streaming environment using Segments (typically CMAF objects), and a manifest such as DASH or HLS, the service provider wants to control the end-to-end latency, possibly also for synchronized playback across the clients, but also wants to measure the latency and report this information back back to the network or service provider.

Latency is typically measure from glass-to-glass or measured from encoder to glass.

Information on the latency may include

* what the actually latency is
* If a desired target latency is met
* The deviation from the target latency
* The reason for the target latency: late arrival, network issues, user controlled, etc.

The network and/or service provider may use this information to

* Do a QoE measure for each client
* Do some network improvements if the latency is not met, e.g. Content Steering, QoS, etc.
* Aggregate the information across multiple/all users

In this document we present the general problem statements and the extensions in DASH that would be needed to support the use case.

## Principles

Assume the architecture below for which the Camera or encoder are synced to a time sync server and so is the DASH client.

A diagram of a software system

AI-generated content may be incorrect.

Then the principle way to solve his:

* *Provide producer reference times*
  + *Add to manifest*
  + *Add prft into CMAF Segments*
* *Synchronize client and server*
  + *External means*
  + *Use UTCTiming*
* *Potentially create a service description for desired latency*
* **Measure the latency in the client**
* **Report the latency/latency offset to the network, for example DASH Metrics or CMCD**
* Use the reports in the network for optimization
* Latency aggregation
* Create network APIs that expose the observed latency.

In DASH, the italics above exist, and only the **bold aspects** are to be addressed.

The principle work flow is as follows

* The DASH client is time-synchronized to a network function that generates a producer reference time and carries this time in the media and the client the uses producer reference time information in the media to measure the latency and reports the observed latency of a media sample or derived information back to the network where the producer reference time is a pair of media time and wall-clock time, and the media time relates to the wall-clock time.
* The details of the measurement is following:
  + The information documented in the producer reference time is used as an anchor and media time is MTA and the wall-clock time is WCA
  + A timescale is known from the media that documents the units per second as TS.
  + Then media time latency MTL of a media time MT presented at wall clock time WC in seconds is determined as MTL = (WC – WCA) - (PT – PTA)/TS
* Example: Let the timecale be set to 20. Assume the media time of a sample being 3740 and the presentation time is presented at 20:18:10.5 and the anchor of the wall-clock time is 20:15:00. Then the presentation latency of the sample is derived as 190.5s - 3740/20 s = 190.5s - 187s = 3.5s.
* The media sample for which the metrics is reported may be
  + The start of the specific segment
  + The start of the previous segment
  + Any media time that is configured in the measurement, for example every 10 seconds
  + Only measured based on events, i.e. the latency is exceeding a threshold
* Not only the latency may be measure, but derived information
  + If a presentation latency is configured (for example by a service description), then the deviation may measured
  + The information may be aggregated and a distribution may be reported
  + The reporting is only initiated if a specific event occurs
  + The reporting may depend on the playback mode of the client, for example in live mode it reports, in timeshift mode not

## What does CMCD v2 define

Note that CMCDv2 defines the following metric (ltc) Live stream latency "The time delta between when a given media timestamp was made available at the origin and when it was rendered by the client. The accuracy of this estimate is dependent on synchronization between the packager and the player clocks. "

CMCDv2 metric is not exactly what we need, as we measure the encoding and end-to-end latency and the definition is more accurate.

## Proposal

It is proposed that

* The metrics are defined
  + Latency: see above
  + latency deviation: the deviation from Target Latency
* the metrics are added to DASH metrics
* We check with CMCD whether they are interested in defining the metric in CMCD or we address this in a DASH metrics calls.
* We may also consider a specific way to carry DASH metrics in CM

“urn:mpeg:dash:…” namespace assignments

## Abstract

Proposal for clarification of URN namespaces defined by DASH family to avoid naming collisions.

## Introduction

According to IETF RFC 3614, URN namespaces for all MPEG developed standards should follow the pattern:

urn:mpeg:{standard name}:{assigned US-ASCII string}

where "{standard name}" is a US-ASCII string that conforms to URN Syntax requirements ([RFC2141]) and corresponds to the name of an MPEG standard (such as "mpeg1", "mpeg2", "mpeg4", "mpeg7", "mpeg21")

Since the "{standard name}" refers to the standard family name and not individual parts thereof, it is common practice for other MPEG standards to designate a sub-namespace for each part of a standard.

An example therefore is

urn:mpeg:mpegB:cicp:

where the 3rd and 4th component refer to:

* mpegB: 🡪 ISO/IEC 23001
* cicp: 🡪 Part 8 of the abovementioned standard

However, with the MPEG DASH standards including all its parts, this concept was not followed and there is no single and obvious source of URN scheme definition.

This implies a potential source of naming collisions.

While Part 1 uses urn:mpeg:dash:, most of all other parts of the MPEG-DASH standards family use this namespace with an additional identifier to span a sub-namespace, e.g.: serverpush: for Part 6 or asset-storage-format: for Part 9.

## Proposal

It is proposed to add the following changes to ISO/IEC 23009-1, clause 4.7 “Schemes”:

| **Scheme identifier** | **Clause in this document** | **Informative description** |
| --- | --- | --- |
|  |  |  |
| … |  |  |
| … |  |  |
|  |  |  |
| **Sub-Namespaces and Scheme defined by other parts** |  |  |
| urn:mpeg:dash:serverpush: | n/a | Namespace used for ISO/IEC 23009-6 |
| urn:mpeg:dash:asset-storage-format: | n/a | Namespace used for ISO/IEC 23009-9 |
| *urn:mpeg:dash:encryptionkey-server* | TBD – see below | TBD |
|  |  |  |

### On urn:mpeg:dash:encryptionkey-server:

This scheme is used in ISO/IEC 23009-9. This scheme does not follow the common naming convention of all other schemes defined by this part of the DASH standards family.

## Recommendations

Since there’s no clear disambiguation for schemes defined by Part 1, this contribution proposes to list all sub-namespace utilized by the DASH standards family in Part 1.

It’s recommended to study the problem of potential naming collisions and consider the proposed text from clause 3 for inclusion into a future version of 23009-1.

Further it’s recommended to check DASH Part 2 to 5 for any other scheme assignments.

## References

* IETF RFC 3614: <https://datatracker.ietf.org/doc/html/rfc3614>

CMCD V1 and V2 (m74567, m75722)

## Introduction

This document addresses CMCD v2 for the next version of DASH. The document is taking into account discussions in the SVTA DASH-IF WG.

Also follow the discussion here: https://github.com/Dash-Industry-Forum/DASH-IF-IOP/issues/440

## Open Questions related to 6th edition

In clause K3.7, client data reporting is provided. However, there are some ambiguities in the definition of this reporting: Client-Data Reporting seemingly permits different reporting schemes. The reporting can seemingly be filtered. However, filtering is unclear whether it refers to

1. The metrics are only measured for data received for a specific service location and/or adaptation set, but reported to all service locations and with every adaptation set request.
2. The metrics are measured for all requests, but the reporting is only enabled for service locations and/or adaptation sets.

Note as it is intended to be used generically, it is suspected that 2 is intended instead of 1. The selected reporting is enabled by the reporting scheme, but then this should be covered under CMCD version 1 specifically, similar to the include request data.

K3.7 Client-Data Reporting Configuration

This document specifies a Metrics reporting scheme in [subclause 5.9](#Section_sec_5.9). However, in several cases an external client data collection is used that needs configuration, including the ability to invoke metadata reporting only for a subset of the MPEG DASH parameters. Reporting may apply only for a subset of DASH requests. A set of filters for configuring a subset of filters is provided in [Table K.7](#Table_tab_K.7).

Table K.7 — Client Metadata Reporting filters

| **Key** | **Type** | **Description** |
| --- | --- | --- |
| serviceLocations | string | specifies a space-delimited list of Service Locations as defined in [subclause 5.6.6](#Section_sec_5.6.6) for which reporting is enabled. If the attribute is absent, then it applies to all Service Locations specified in the MPD. |
| adaptationSets | string | specifies a space-delimited list of identifiers @id of Adaptation Sets for which reporting is applied. If the attribute is absent, then it applies to all eligible Adaptation Sets.  Service Locations and Adaptation Sets filters are applied jointly addressing the intersection of the two. |
| **ReportingSystem** | descriptor | specifies the reporting system and configuration to send back client data for the intersection of the Service Locations and Adaptation Sets, if both are provided. |

A specific client data reporting system is Common Media Client Data (CMCD) specified in CTA-5004.

A descriptor is defined as urn:mpeg:dash:cta-5004:2023 and the value of the descriptor is the version number of the specification. [Table K.8](#Table_tab_K.8) provides relevant parameters to configure CMCD reporting.

Table K.8 — CMCD specific parameters

| **Key** | **Type** | **Description** |
| --- | --- | --- |
| version | unsigned int | specifies the highest *CMCD version* as defined in CTA-5004 that is accepted by the reporting server.  If absent, the version is assumed to be version 1 as defined in CTA-5004. |
| mode | string | specifies the data transition mode how the media client shall send the media client data as defined in CTA-5004, Clause2.  The two options are "query" and "header". "header" refers to the mode defined in CTA-5004, subclause 2.1. "query" refers to the mode defined in CTA-5004, subclause 2.2.  If the value is absent, the "query" method shall be used.  NOTE The third method, including the data in a JSON object, is not defined in this document as CTA-5004 does not define a detailed protocol for using it. |
| includeInRequests | string | specifies which HTTP GET requests that shall include CMCD data. The semantics are identical to the @includeInRequests attribute in [Table I.3](#Table_tab_I.3). |
| keys | string | specifies a space-delimited list of the CMCD keys which shall be reported as a part of CMCD reporting. Values of the keys are listed in CTA-5004.  Keys not defined in CTA-5004 shall not be present in the attribute unless they use the custom key syntax defined in CTA 5004. Unsupported keys shall be ignored by the client and not reported. |
| contentId | string | specifies the value of the *Content ID*cid key in CMCD reporting, if cid is present in the keys attribute. The value shall be restricted to maximum of 64 characters.  If the value is absent or is invalid and the *Content ID* needs to be reported, then the value of *Content ID* is derived as follows:  If there exists an **up:ExtUrlQueryInfo** element with an @includeInRequests attribute having value of "urn:mpeg:dash:cmcd#cid", the *Content ID* value shall be the value of the corresponding *finalQueryString* derived using the process defined in Clause [I.3](#Annex_sec_I.3).  If the above does not result in a valid *Content ID* value, the client shall not report the cid key. |
| sessionID | string | specifies the value of the *Session ID*sid key in CMCD reporting, if sid is present in the keys attribute. The value shall be restricted to maximum 64 characters and the authors should make sure it complies with the definitions in CTA 5004.  It is not recommended to provide *Session ID* in the MPD, unless the author intends this MPD to be unique per client.   1. If the value is absent or is invalid and the *Session ID* needs to be reported, then the value of *Session ID* is derived as follows:If there exists an **up:ExtUrlQueryInfo** element with an @includeInRequests attribute having value of "urn:mpeg:dash:cmcd#sid", the *Session ID* value shall be the value of the corresponding *finalQueryString* derived using the process defined in Clause [I.3](#Annex_sec_I.3) 2. In case a *SessionID* value could not be derived as specified above, the DASH client shall generate a UUID formatted Session ID per RFC 9562 and use the same sid for the entire session. Use of UUIDv7 for this purpose is recommended to simplify processing of CDN logs. |

A proposed fix is as follows:

* Separate three aspects
  1. Filter for what is reported 🡺 owned by DASH
  2. Filter what is reported 🡺 owned by reporting system, like CMCD
  3. Filter how it is reported 🡺 owned by reporting system, also CMCD
  + Note that logically, 1 and 2 could be combined as they are a filter on the measurement (which metrics and for what)
  + However, as CMCD does both, this is not trivial.

For 1, DASH today defines service locations and Adaptation Sets. This could be generalized further, for example for Periods, for specific media types, and so on.

For 2 and 3, this needs to added to the reporting system. This can be updated.

Proposed updates are provided.

K3.7 Client-Data Reporting Configuration

This document specifies a the configuration of Metrics reporting scheme in [subclause 5.9](#Section_sec_5.9). However, in several cases an external client data collection may be used that needs configuration, including the ability to invoke metadata reporting only for a subset of the DASH Media Presentation.

In addition, system itself and the

The configuration of a reporting system is provided in [Table K.7](#Table_tab_K.7).

Table K.7 — Client Metadata Reporting filters

| **Key** | **Type** | **Description** |
| --- | --- | --- |
| serviceLocations | string | specifies a space-delimited list of Service Locations as defined in [subclause 5.6.6](#Section_sec_5.6.6) for which the metrics are collected. If the attribute is absent, then it applies to all Service Locations specified in the MPD. |
| adaptationSets | string | specifies a space-delimited list of identifiers @id of Adaptation Sets for which the metrics are collected. If the attribute is absent, then it applies to all eligible Adaptation Sets.  Service Locations and Adaptation Sets filters are applied jointly addressing the intersection of the two. |
| **ReportingSystem** | descriptor | specifies the reporting system and configuration to send back client data for the intersection of the Service Locations and Adaptation Sets, if both are provided. |

A specific client data reporting system is Common Media Client Data (CMCD) specified in CTA-5004.

A descriptor is defined as urn:mpeg:dash:cta-5004:2023 and the value of the descriptor is the version number of the specification. [Table K.8](#Table_tab_K.8) provides relevant parameters to configure CMCD reporting.

Table K.8 — CMCD specific parameters

| **Key** | **Type** | **Description** |
| --- | --- | --- |
| mode | string | specifies the data transition mode how the media client shall send the media client data as defined in CTA-5004, Clause2.  The two options are "query" and "header". "header" refers to the mode defined in CTA-5004, subclause 2.1. "query" refers to the mode defined in CTA-5004, subclause 2.2.  If the value is absent, the "query" method shall be used.  NOTE The third method, including the data in a JSON object, is not defined in this document as CTA-5004 does not define a detailed protocol for using it. |
| includeInRequests | string | specifies which HTTP GET requests that shall include CMCD data. The semantics are identical to the @includeInRequests attribute in [Table I.3](#Table_tab_I.3). |
| includeInServiceLocation | string | specifies a space-delimited list of Service Locations as defined in [subclause 5.6.6](#Section_sec_5.6.6) for which the requests shall include CMCD data.  If the attribute is absent, then it applies to all Service Locations specified in the MPD. |
| includeInAdaptationSets |  | specifies a space-delimited list of Adaptation Sets by the value of @id for which the requests shall include CMCD data.  Request Types, Service Locations and Adaptation Sets filters are applied jointly addressing the intersection of the two. |
| keys | string | specifies a space-delimited list of the CMCD keys which shall be reported as a part of CMCD reporting. Values of the keys are listed in CTA-5004.  Keys not defined in CTA-5004 shall not be present in the attribute unless they use the custom key syntax defined in CTA 5004. Unsupported keys shall be ignored by the client and not reported. |
| contentId | string | specifies the value of the *Content ID*cid key in CMCD reporting, if cid is present in the keys attribute. The value shall be restricted to maximum of 64 characters.  If the value is absent or is invalid and the *Content ID* needs to be reported, then the value of *Content ID* is derived as follows:  If there exists an **up:ExtUrlQueryInfo** element with an @includeInRequests attribute having value of "urn:mpeg:dash:cmcd#cid", the *Content ID* value shall be the value of the corresponding *finalQueryString* derived using the process defined in Clause [I.3](#Annex_sec_I.3).  If the above does not result in a valid *Content ID* value, the client shall not report the cid key. |
| sessionID | string | specifies the value of the *Session ID*sid key in CMCD reporting, if sid is present in the keys attribute. The value shall be restricted to maximum 64 characters and the authors should make sure it complies with the definitions in CTA 5004.  It is not recommended to provide *Session ID* in the MPD, unless the author intends this MPD to be unique per client.   1. If the value is absent or is invalid and the *Session ID* needs to be reported, then the value of *Session ID* is derived as follows:If there exists an **up:ExtUrlQueryInfo** element with an @includeInRequests attribute having value of "urn:mpeg:dash:cmcd#sid", the *Session ID* value shall be the value of the corresponding *finalQueryString* derived using the process defined in Clause [I.3](#Annex_sec_I.3) 2. In case a *SessionID* value could not be derived as specified above, the DASH client shall generate a UUID formatted Session ID per RFC 9562 and use the same sid for the entire session. Use of UUIDv7 for this purpose is recommended to simplify processing of CDN logs. |

The equivalent updates need to be done for Annex K.4.2.7

Note that there are more inconsistencies in the reporting system, the URN for the reporting system is not even used. Also it allows to report N times, which also needs clarification. Also this semantic hints clearly towards the interpretation from above, namely that collection is

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **CMCDParameters** | 0 … N | Defines reporting system parameters to send back client data for the above Service Locations and Adaptation Sets for CMCD. For details refer to [subclause K.4.2.7.2](#Annex_sec_K.4.2.7.2). |

## What about CMCDv2

dash.js has prepared a configuration object to configure reporting.

<https://github.com/Dash-Industry-Forum/dash.js/pull/4816/files?diff=unified&w=0>

A clean, valid JSON configuration for CMCD reporting is provided.

{  
 "cmcd": {  
 "version": 2,  
 "enabled": true,  
 "sid": "b248658d-1d1a-4039-91d0-8c08ba597da5",  
 "cid": "21cf726cfe3d937b5f974f72bb5bd06a",  
 "mode": "CMCD\_MODE\_QUERY",  
 "enabledKeys": [  
 "br", "d", "ot", "tb", "bl", "dl", "mtp", "nor", "nrr", "su",  
 "bs", "rtp", "cid", "pr", "sf", "sid", "st", "v", "msd"  
 ],  
 "targets": [  
 {  
 "enabled": true,  
 "url": "http://localhost:3001/cmcd/response-received",  
 "enabledKeys": ["url", "rc", "msd"],  
 "includeOnRequests": ["mpd", "segment"],  
 "events": ["rr"],  
 "mode": "CMCD\_MODE\_QUERY"  
 },  
 {  
 "enabled": true,  
 "url": "http://localhost:3002/cmcd/event-mode",  
 "timeInterval": 10,  
 "enabledKeys": ["e", "msd", "sta"],  
 "events": ["ps", "e", "t"],  
 "mode": "CMCD\_MODE\_QUERY"  
 },  
 {  
 "enabled": true,  
 "url": "http://localhost:3003/cmcd/event-mode",  
 "timeInterval": 6,  
 "mode": "CMCD\_MODE\_HEADER"  
 },  
 {  
 "enabled": true,  
 "url": "http://localhost:3004/cmcd/event-body-mode",  
 "mode": "CMCD\_MODE\_BODY",  
 "batchSize": 3  
 },  
 {  
 "enabled": true,  
 "url": "http://localhost:3005/cmcd/event-body-mode",  
 "mode": "CMCD\_MODE\_BODY",  
 "batchTimer": 3  
 }  
 ]  
 }  
}

An overview of the semantics is provided.

**Global CMCD settings**

* version: 2 — Uses CMCD v2 key semantics.
* enabled: true — CMCD reporting is active for requests/events.
* mode: CMCD\_MODE\_QUERY — Default CMCD transport is query-string unless overridden per target.
* sid and cid — Session and Content IDs included everywhere for correlation.

**Enabled keys (global)**

* Emitted across requests/events when applicable: br, d, ot, tb, bl, dl, mtp, nor, nrr, su, bs, rtp, cid, pr, sf, sid, st, v, msd.

**Targets (per-destination transport & schedule)**

1. Response-received events (rr) — Query mode; sends url, rc, msd on MPD & segment requests to http://localhost:3001/...
2. Event mode (query) — Every 10s, emits events ps, e, t with e, msd, sta to http://localhost:3002/...
3. Event mode (header) — Every 6s, attaches CMCD as HTTP headers to http://localhost:3003/...
4. Event body mode (batched by count) — Body mode; batch sends after collecting 3 events to http://localhost:3004/...
5. Event body mode (batched by timer) — Body mode; batch sends every 3s to http://localhost:3005/...

### Example Emissions

**3.1 Query-Mode on MPD Request**

Request URL (MPD fetch)

GET https://example.com/manifest.mpd?CMCD=br%3D4800%2Cot%3Dm%2Ctb%3D4500%2Cbl%3D12000%2Cdl%3D3000%2Csu%2Csf%3Dd%2Csid%3Db248658d-1d1a-4039-91d0-8c08ba597da5%2Ccid%3D21cf726cfe3d937b5f974f72bb5bd06a%2Cv%3D2%2Cmsd%3D123

rr Event to Target 1 (query mode)

GET http://localhost:3001/cmcd/response-received?CMCD=url%3Dhttps%3A%2F%2Fexample.com%2Fmanifest.mpd%2Crc%3D200%2Cmsd%3D124

**3.2 Segment Request with CMCD in Query**

Request URL (Segment fetch)

GET https://cdn.example.com/video/seg-1200.m4s?CMCD=br%3D1200%2Cot%3Da%2Ctb%3D2000%2Cbl%3D8000%2Cdl%3D2500%2Crtp%3D1%2Cst%3Dl%2Csid%3Db248658d-1d1a-4039-91d0-8c08ba597da5%2Ccid%3D21cf726cfe3d937b5f974f72bb5bd06a%2Cv%3D2%2Cmsd%3D130

rr Event to Target 1 (query mode)

GET http://localhost:3001/cmcd/response-received?CMCD=url%3Dhttps%3A%2F%2Fcdn.example.com%2Fvideo%2Fseg-1200.m4s%2Crc%3D206%2Cmsd%3D131

**3.3 Periodic Events in Header Mode (every 6s)**

GET http://localhost:3003/cmcd/event-mode  
**CMCD-Object:** br=2400,ot=v,tb=3200,bl=9000,dl=2000  
**CMCD-Session:** su,sid=b248658d-1d1a-4039-91d0-8c08ba597da5,cid=21cf726cfe3d937b5f974f72bb5bd06a  
**CMCD-Request:** sf=d,v=2,msd=140

**3.4 Event Mode in Query (every 10s)**

GET http://localhost:3002/cmcd/event-mode?CMCD=e%3Dps%2Cmsd%3D150%2Csta%3Dstartup

GET http://localhost:3002/cmcd/event-mode?CMCD=e%3Dt%2Cmsd%3D152%2Csta%3Dtrick

**3.5 Event Body Mode, Batched by Size**

POST http://localhost:3004/cmcd/event-body-mode  
**Content-Type:** application/json  
  
{  
 "batch": [  
 { "e": "ps", "msd": 160, "sta": "startup", "sid": "b248658d-1a-4039-91d0-8c08ba597da5", "cid": "21cf726cfe3d937b5f974f72bb5bd06a", "v": 2 },  
 { "e": "e", "msd": 162, "sta": "rebuffer", "sid": "b248658d-1d1a-4039-91d0-8c08ba597da5", "cid": "21cf726cfe3d937b5f974f72bb5bd06a", "v": 2 },  
 { "e": "t", "msd": 165, "sta": "seek", "sid": "b248658d-1d1a-4039-91d0-8c08ba597da5", "cid": "21cf726cfe3d937b5f974f72bb5bd06a", "v": 2 }  
 ]  
}

**3.6 Event Body Mode, Batched by Timer (every 3s)**

POST http://localhost:3005/cmcd/event-body-mode  
**Content-Type:** application/json  
  
{  
 "batch": [  
 { "e": "ps", "msd": 170, "sta": "startup", "sid": "b248658d-1d1a-4039-91d0-8c08ba597da5", "cid": "21cf726cfe3d937b5f974f72bb5bd06a", "v": 2 }  
 ]  
}

**JSON Schema**  
{  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "$id": "https://example.com/schemas/cmcd-config.schema.json",  
 "title": "CMCD Configuration",  
 "type": "object",  
 "required": ["cmcd"],  
 "properties": {  
 "cmcd": {  
 "type": "object",  
 "required": ["version", "enabled", "mode", "sid", "cid", "enabledKeys", "targets"],  
 "properties": {  
 "version": {  
 "type": "integer",  
 "enum": [1, 2],  
 "description": "CMCD spec version to apply to key semantics."  
 },  
 "enabled": {  
 "type": "boolean",  
 "description": "Enable CMCD reporting globally."  
 },  
 "sid": {  
 "type": "string",  
 "format": "uuid",  
 "description": "Session ID; invariant over the life of the playback session."  
 },  
 "cid": {  
 "type": "string",  
 "description": "Content ID; stable across sessions/devices for the same title/asset."  
 },  
 "mode": {  
 "type": "string",  
 "enum": ["CMCD\_MODE\_QUERY", "CMCD\_MODE\_HEADER", "CMCD\_MODE\_BODY"],  
 "description": "Default transport for CMCD data unless overridden per target."  
 },  
 "enabledKeys": {  
 "type": "array",  
 "items": { "type": "string" },  
 "description": "Whitelist of CMCD keys allowed to be emitted by the client."  
 },  
 "targets": {  
 "type": "array",  
 "minItems": 1,  
 "items": {  
 "type": "object",  
 "required": ["enabled", "url", "mode"],  
 "properties": {  
 "enabled": {  
 "type": "boolean",  
 "description": "Enable this target destination."  
 },  
 "url": {  
 "type": "string",  
 "format": "uri",  
 "description": "Destination endpoint for CMCD emissions."  
 },  
 "mode": {  
 "type": "string",  
 "enum": ["CMCD\_MODE\_QUERY", "CMCD\_MODE\_HEADER", "CMCD\_MODE\_BODY"],  
 "description": "Transport for CMCD when emitting to this target."  
 },  
 "enabledKeys": {  
 "type": "array",  
 "items": { "type": "string" },  
 "description": "Target-specific whitelist of keys (overrides global if present)."  
 },  
 "includeOnRequests": {  
 "type": "array",  
 "items": {  
 "type": "string",  
 "enum": ["mpd", "segment", "xlink", "steering", "other"]  
 },  
 "description": "Request types where CMCD is included for this target."  
 },  
 "events": {  
 "type": "array",  
 "items": {  
 "type": "string",  
 "enum": ["rr", "ps", "e", "t"]  
 },  
 "description": "Event types to emit to this target (rr=response-received, ps=playback-start, e=error, t=trick)."  
 },  
 "timeInterval": {  
 "type": "integer",  
 "minimum": 1,  
 "description": "Periodic emission interval in seconds (for event/heartbeat style reporting)."  
 },  
 "batchSize": {  
 "type": "integer",  
 "minimum": 1,  
 "description": "Batch N events before sending (BODY mode only)."  
 },  
 "batchTimer": {  
 "type": "integer",  
 "minimum": 1,  
 "description": "Flush batch every N seconds (BODY mode only)."  
 }  
 },  
 "allOf": [  
 {  
 "if": { "properties": { "mode": { "const": "CMCD\_MODE\_BODY" } } },  
 "then": {  
 "anyOf": [  
 { "required": ["batchSize"] },  
 { "required": ["batchTimer"] }  
 ]  
 }  
 }  
 ]  
 }  
 }  
 }  
 }  
 }  
}

Configuration fields (Global):

* **version:** Integer (1 or 2). CMCD spec revision governing key semantics.
* **enabled:** Boolean. Master switch to enable/disable CMCD emissions.
* **sid:** String (UUID). Session ID; constant for the playback session.
* **cid:** String (<=64 chars recommended). Content ID; stable across sessions/devices.
* **mode:** Enum: CMCD\_MODE\_QUERY | CMCD\_MODE\_HEADER | CMCD\_MODE\_BODY. Default transport for CMCD.
* **enabledKeys:** Array<string>. Global whitelist of CMCD keys allowed to emit.
* **targets:** Array<Target>. Per-destination configuration including transport, events, batching.

Target fields

* **enabled:** Boolean. Enables this endpoint.
* **url:** URI. Destination endpoint for CMCD.
* **mode:** Enum. Transport override for this target.
* **enabledKeys:** Array<string>. Narrow per-target key set.
* **includeOnRequests:** Array<mpd|segment|xlink|steering|other>. Request types to include CMCD.
* **events:** Array<rr|ps|e|t>. rr=response-received; ps=playback-start; e=error; t=trick-play.
* **timeInterval:** Integer (s). Periodic emission interval.
* **batchSize:** Integer. Batch N events before sending (BODY mode).
* **batchTimer:** Integer (s). Flush batch every T seconds (BODY mode).

**Transport semantics**

* Header mode: keys split across CMCD-Object, CMCD-Request, CMCD-Session (and Status). Use comma-separated tokens; flags without values.
* Query mode: single CMCD= parameter with comma-separated tokens (URL-encoded). Do not mix header and query in the same request.
* Body mode: some collectors/validators support JSON payloads, often batched to reduce overhead.

**Example header & query formatting**

GET https://example.com/segment.m4s  
**CMCD-Object:** br=2400,ot=v,d=4000,bl=9000  
**CMCD-Request:** sf=d,v=2  
**CMCD-Session:** sid=b248658d-1d1a-4039-91d0-8c08ba597da5,cid=21cf726cfe3d937b5f974f72bb5bd06a  
**CMCD-Status:** su,bs

GET https://cdn.example.com/segment.m4s?CMCD=br%3D2400%2Cot%3Dv%2Cd%3D4000%2Cbl%3D9000%2Csf%3Dd%2Cv%3D2%2Csid%3Db248658d-1d1a-4039-91d0-8c08ba597da5%2Ccid%3D21cf726cfe3d937b5f974f72bb5bd06a%2Csu%2Cbs

Now based on all of these examples, some the CMCD configuration makes it more explicit to have

* What keys to collect
* Allowing different targets for reporting

Energy-aware Streaming (m75723)

## Introduction

Service-related information related to energy consumption is not addressed in streaming services yet. As an example, 3GPP and Greening of Streaming look at some aspects related to streaming services

This proposal provides initial discussion on the following:

* Integrate energy-related information into general media streaming services.
* Create an extensible framework that can be used by different environments.
* Combine the general framework with 3GPP services.

## Functional description

### Overview

Figure X provides an overview of a generic streaming workflow. The workflow may be considered as simple, and can obviously be extended for redundant operations, adding DRM, inserting advertisements, and so on. However, for the purpose of the discussion, this is not relevant.

A diagram of a computer

AI-generated content may be incorrect.

Figure X: Streaming architecture with energy related information workflows

The workflow is as follows:

1. Produced content is provided to a distribution encoder, possibly in multiple variants (for example different resolutions, an SDR and an HDR version, and so on). The captured content may have associated static and/or dynamic information related to the energy needed to produce the content.
2. The encoder encodes the content and aggregates the energy information provided upstream, and the energy/power that it required to encode, possibly also for different variants, for example the encoding technology, the bit rate and so on.
3. The aggregated energy information may then be propagated through the distribution chain, by aggregating and accumulating the energy usage at each stage of the distribution.

A core concept is that the aggregated energy information is included as metadata in the media streaming data, and different options are possible. The media may be delivered via a 3GPP delivery network and reach a media client, but it may also be delivered via other means that are not 3GPP-based. The media client is typically not 3GPP-specific and hence information is expected to be agnostic of the media delivery system and access network(s) used to deliver it. The media client or any node in the network may use information provided in the metadata to optimize the delivery based on energy-related metrics.

In summary, the following functional aspects are considered:

1. Propagate energy-related information downstream within the media streaming workflow
2. Make the energy-related information accessible at the level of individual media streams, such that nodes in the delivery chain can access the information.
3. Allow the energy-related information to be aggregated and/or accumulated at specific nodes in the media delivery path.
4. Collect energy-related information at appropriate nodes in the media delivery path and use the provided information for operational decision-making.
5. Allow the aggregated and accumulated energy-related information to be reported to reporting servers:
6. Excluding the measured energy consumption at the client.
7. Including the measured energy consumption at the client.
8. Allow the client to select content based on energy-related metrics.

In order to provide meaningful and trustworthy energy reporting, the nodes in the media delivery chain adding energy-related metrics are preferably trusted energy consumption reporters.

In the following, some specific aspects are discussed.

### Assignment and aggregation of energy usage

Media resources available on a specific storage or distribution node may include a record of the energy that has been consumed up to the current point in the media delivery chain, for example including production, encoding, distribution and so on. The information may be scaled to the number of users assigned to this resource. Such a scaling is obviously difficult to estimate, but for example for a Segment, it may be predicted how many users were requesting the previous Segment in the same Representation, and hence such information can be extrapolated. Aggregation may be done by summing the energy information or by other means of aggregation, depending on the assigned metrics.

The energy metric may be assigned to the entire media asset, or to a subset of the media asset, for example to a specific variant, to a specific segment or to a time range.

### Common repository of energy-related metrics

Metrics assigned to energy consumption are currently not well defined, but may improve over time. A framework is expected to be applicable with an evolution of energy metrics. For this purpose it is proposed to create a registry/repository/catalogue for energy-related metrics, similar to MP4RA (https://mp4ra.org/) or DASH-IF identifiers (https://dashif.org/identifiers/introduction/). To each metric a key may be added, for example using a 4CC code or a URN. In addition, the metric is well-defined, typically by reference to a specification and/or implementation. This allows different metrics to be used, and also avoids the need for frameworks to redundantly define the metrics they use. As hypothetical examples:

- Greening of Streaming defines a URN urn:org:gos:energyindex:2025 and/or a 4CC 'eeix' with a well-defined value assigned to it and the energy index expressing the average aggregated consumed energy per second for the media assigned to it.

- French regulator AFNOR defines urn:org:afnor:carbonindex:2025 and/or a 4CC 'eecx' with a well-defined value being assigned to it and the carbon index expresses the average aggregated consumed carbon per second for the media assigned to it.

Such a repository may be hosted by a registration authority, or for example by an organization such as Greening of Streaming.

### In-band carriage of energy-related metrics

Once the metrics are well defined, for example following what is presented in clause 7.x.2.3, the information may be carried by different means within the media delivery chain. Examples include, but are not limited to:

- *Metadata tracks:* If the information is dynamic, a metadata track may be used to carry one or different energy-related metrics. Metrics may also be delivered, for example, as part of Addressable Resource Index (ARI) tracks as defined in ISO/IEC 23009-1 [23009-1].

- *Common Media Client Data (CMCD) metrics:* In the reporting mechanism associated with media delivery, custom energy metric keys may be defined, for example using the definitions in clause 7.x.2.3, and providing the energy related to the consumed assets.

- *Common Media Server Data (CMSD) information:* In the streaming of media assets, information may be assigned to each resource using data for each segment. This may again use custom energy metric keys.

- *SAND information:* CMCD- and CMSD-equivalent information may also be provided as part of SAND messages as defined in ISO/IEC 23009-5 [23009-5], for example using custom keys.

- *DASH metrics:* Equivalently to CMCD, DASH metrics as defined in ISO/IEC 23009-? [23009-?] may be used to report energy-related information.

- *Events and Event Streams: Event Streams* as defined in ISO/IEC 23009-? [23009-?] may carry energy-related information, similarly to CMSD.

- *Information in streaming presentation manifest:* A DASH MPD may carry information about energy metrics, for more details refer to clause 7.x.2.5. Equivalently to the information in a DASH MPD, energy metrics may be carried in other streaming manifest formats such as HLS playlists.

- Service Information and API calls as defined in MPEG DASH allow operational information to be carried that defines the API to the client from the application and may provide energy-related information.

The information may be assigned to the entire media delivery chain or parts of the chain.

### Carriage of energy-related information in streaming presentation manifest

A streaming presentation manifest may carry energy-related information. In the following, examples for a DASH MPD are provided.

- Energy information may be assigned to each resource in the MPD, including all information relevant for the service provider.

- Energy information may be assigned to individual components of the presentation:

- Content – providing information on aggregate energy consumption

- An Adaptation Set

- A Representation

- A Segment

- Service Locations – providing information on the energy required to distribute content from this service location, possibly normalized to the bit rate.

- Adding labels describe energy-related information. Labels are for example available in DASH MPDs to expose information to the user for user-based selection of resources.

- In order to address regulatory requirements, a dedicated Energy descriptor **Energy** is added to the DASH manifest that allows carriage and exposure of energy-related information, similar to Accessibility information.

- If information is available on different levels, it may be combined (summed, for example, or combined after scaling appropriately). As an example, in a Manifest, energy related information is associated to a BaseURL or a service location describing the CDN energy efficiency scaled with bitrate, but is also assigned to a Representation describing the encoding of the resources. In this case the consumption of this resource is a combination of using this resource and this CDN and the client may combine this to report overall energy.

### Usage of energy information in media client or other functions

The information in the presentation manifest, as documented in clause 7.x.2.5 may be used in the media client for different means:

- Exposing energy-related information to the user.

- Exposing energy-related information to the application.

- Based on the above, the user and/or application can use the energy-related information provided to select content based on energy. Obviously such operational changes may change the actual information, and users may be informed on such updates, for examples using CMSD headers.

- Selecting content and/or service locations based on energy-related information.

- Reporting the aggregated metric to a reporting server, for example using DASH Metrics or CMCD.

- Possibly adding client energy consumption to the aggregated metrics.

Other functions in the media delivery chain may use the in-band information for operational and reporting purposes as well.

## Procedures

Figure 2 provides a general high-level call flow for enery-aware streaming.

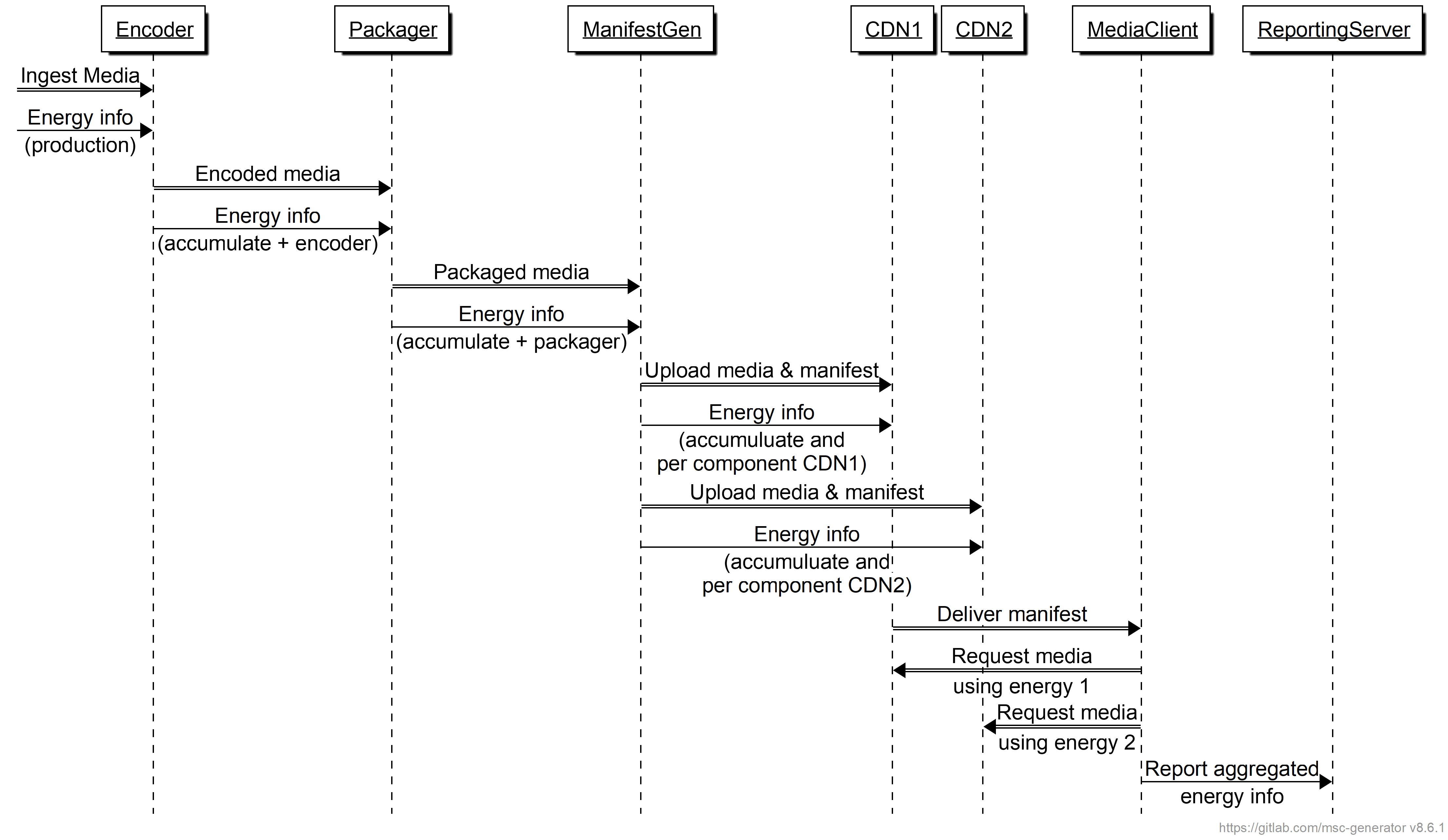


Figure 7.X.3.1-1 General call flow for enery-aware streaming

Here is the description of the call flow:

1. Encoder

a) Processes ingested media.

b) processes energy information (production).

c) Sends encoded media to Packager.

d) Shares accumulated energy info with Packager.

2. Packager

a) Receives encoded media.

b) Packages media.

c) Sends packaged media to Manifest Generator.

d) Shares accumulated energy info with Manifest Generator.

3. Manifest Generator

a) Receives packaged media.

b) Uploads media and manifest to CDN1 and CDN2.

c) Adds energy info for each media component (Adaptation Set, Representation, etc.).

d) Adds energy information for each service location/CDN.

4. Delivery

a) Delivers manifest to Media Client.

b) Media Client request media based on energy information from CDN1 and/or CDN2.

5. Reporting

- Media client reports aggregated energy information to Reporting Server.